VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Aplikace pro správu portfolia a vizualizaci finančních trhů

Application for Portfolio Management and Stock Market Visualisation

2012 **Bc. Miroslav Boublík**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Miroslav Boublík**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Aplikace pro správu portfolia a vizualizaci finančních trhů
Application for Portfolio Management and Stock Market Visualisation

Zásady pro vypracování:

Tématem diplomové práce je aplikace pro správu portfolia více uživatelů a vizualizaci finančních trhů. Data budou přístupná jak z webu, tak i z aplikace běžící na mobilních zařízení s operačním systémem Windows Phone 7 (případně WM6.5).

1. Implementace serverové části, která bude zpracovávat veřejně dostupná data (Yahoo Finance, Google Finance apod.), pokud možno v co nejkratším časovém intervalu.
2. Webové aplikace s přihlašováním jednotlivých uživatelů. Zde budou mít uživatelé možnost zpravovat své vlastní portfolio (včetně grafů, nejběžnějších ukazatelů a dalších informací).
3. Aplikace pro mobilní telefony, která bude taktéž umožňovat po přihlášení zobrazení svého portfolia (které bude srovnatelné s verzi z webové aplikace, ovšem s menším množstvím informací z důvodu pomalejších datových přenosů).

Seznam doporučené odborné literatury:

Charles Petzold, Microsoft Silverlight Edition: Programming Windows Phone 7, Microsoft Press; 1 edition, 2010, ISBN 978-0735656673
Nick Randolph, Professional Windows Phone 7 Application Development: Building Applications and Games Using Visual Studio, Silverlight, and XNA, Wrox; 1 edition, 2010, ISBN 978-0470891667
Anatoly B. Schmidt, Financial Markets and Trading: An Introduction to Market Microstructure and Trading Strategies, Wiley; 1 edition, 2011, ISBN 978-0470924129

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michal Krumnikl**

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 30.4.2012

.....

Bc. Miroslav Boublík

Poděkování

Rád bych na tomto místě poděkoval svému vedoucímu, Ing. Michalu Krumniklovi, za odbornou pomoc a konzultaci, kterou mi poskytnul při vytváření této diplomové práce.

Abstrakt

Tato diplomová práce se zabývá sledováním a vizualizací vývoje akcií na akciových burzách. Cílem bylo vytvořit tři aplikace. První aplikací je aplikace běžící na serveru, která má za úkol vytvořit databázi a periodicky ji plnit daty z akciových trhů staženými z veřejně dostupných zdrojů (Yahoo Finance, Google Finance apod.). Druhou aplikací je webový klient, který využívá databázi a umožňuje uživateli spravovat své vlastní portfolio akcií včetně zobrazení průběhu vývoje akcie pomocí grafů. A nakonec mobilní klient, který sdílí klíčové funkce s webovým klientem, ale některé funkce jsou omezeny z důvodu datově omezené, či pomalé, datové sítě. Text samotný se zkráceně zabývá popisem teorie obchodování na burze, průzkumem trhu s existujícími aplikacemi, přibližuje použité technologie a popisuje samotnou implementaci včetně popisu klíčových tříd a programu samotného. Na závěr je provedeno testování a shrnuty jeho výsledky.

Klíčová slova

Akcie, Grafy, Portfolio, Obchodování na burze, MS SQL, .NET, ASP.NET, Windows Phone 7, Silverlight, Klient, Server

Abstract

This thesis deals with monitoring and visualization of the stock market shares. The aim was to create three applications. The first application is a desktop application running on the server, which aims to create a database and periodically fills the database with downloaded stock data from publicly available sources (Yahoo Finance, Google Finance etc.). Another application is a web client that uses a database and allows users to manage their own stock portfolio, including graphs, which are used for visualization of development of the stock market. And finally the mobile client application, which shares key features with the web client, but some functions are limited due to data-limited or slow data network. The text itself presents short description of the theory of stock trading, market research with existing applications, introduces and describes the used technology and describes the actual implementation of the core classes and the program itself. Finally, the testing is done and the results of testing are summarized.

Key words

Shares, Graphs, Portfolio, Stock trading, MS SQL, .NET, ASP.NET, Windows Phone 7, Silverlight, Client, Server

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
OHLC	Open High Low Close	Otevírací nejvyšší nejnižší zavírací
SMA	Simple moving average	Jednoduchý klouzavý průměr
EMA	Exponential moving average	Exponenciální klouzavý průměr
API	Application Programming Interface	Rozhraní pro programování aplikací
GUI	Graphical User Interface	Grafické uživatelské rozhraní
USD	United States dollar	Americký dolar

Seznam použitých termínů

Termín	Význam termínu
Wrapper	Mezivrstva, která v případech uvedených v této práci umožňuje volat neřízený (unmanaged) kód z kódu řízeného (managed)
API	Je sbírkou procedur, funkcí či tříd, které může programátor využít. API určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu.
Windows API	Je API vyvinuté firmou Microsoft pro operační systém Windows. Všechny programy ve Windows musí komunikovat prostřednictvím Windows API, které obsahuje nejen základní funkce, ale i funkce pro vytváření uživatelského rozhraní atd.
XML	Neboli eXtended Markup Language, je značkovací jazyk vyvinutý konsorciem W3C. Jazyk je určen především pro přenos dat a jejich ukládání.
NASDAQ	Největší ryze elektronická akciová burza v USA.

Obsah

1 Úvod

Pro zobrazení a vizualizaci vývoje akcií na akciových trzích po celém světě existuje široké spektrum aplikací. Aplikace zdarma většinou bývají napojeny na veřejně dostupná data, která sice nejsou úplně aktuální, ale na druhou stranu poskytují vše co je potřeba. Problém může nastat, pokud mají být na tyto zdroje napojeny mobilní aplikace. Většinou je potřeba tato data stáhnout do mobilního zařízení, poté zpracovat a až poté je lze zobrazit. Navíc pokud takto definovaný zdroj dat přestane fungovat, přestane fungovat také mobilní aplikace. Toto ovšem většinou nehrozí u placených aplikací, kde jsou data jednak aktuální a jednak odpadá nutnost zpracování dat, neboť stažená data jsou již v podobě vhodné pro zobrazení.

Cílem této diplomové práce je vytvořit aplikaci, která bude využívat data z libovolných bezplatných veřejně dostupných zdrojů, zpracovávat tato data a ukládat je do databáze. Tuto databázi budou poté využívat dva klienti – jeden webový klient a jeden mobilní klient. Oba klienti budou dostupní po registraci a přihlášení, budou poskytovat jednoduchou správu portfolia akcií, s možností vedení údajů o nákupech a prodejích akcií včetně výpočtu zisku a převodu zisku na jinou měnu dle zadaného kurzu. Dále budou poskytovat možnost zobrazení grafů včetně zobrazení základních technických ukazatelů s možností přidat ukazatele vlastní. Mobilní aplikace bude mít některé funkce omezeny z důvodu datově omezené či pomalé mobilní sítě. Jako platformu pro mobilní aplikaci jsem zvolil zatím méně rozšířený systém Windows Phone 7 a to hlavně z toho důvodu, že na tento systém zatím existuje málo bezplatných aplikací tohoto typu.

Díky možnosti změny zdroje dat plnění databáze nedojde k tomu, že aplikace nebudou fungovat v případě nedostupnosti původního zdroje dat a taktéž pokud se objeví zdroj dat, který bude zdarma poskytovat data v reálném čase, můžou jej aplikace bez problému využít. Tímto jsou v podstatě tyto aplikace, co se využití dat týče, přiblíženy jejich placeným verzím.

2 Teorie obchodování na burze

V této kapitole se budu věnovat zjednodušeně teorii obchodování na burze, tedy hlavním informacím, které jsou nutné pro porozumění chodu později popisovaných aplikací. Tato kapitola je rozdělena na dvě části a to na portfolio a grafy.

2.1 Portfolio

Každý obchodník s akciemi si potřebuje vést informace o svých nakoupených či prodaných akciích. Většinou vyžaduje kromě vývoje cen jednotlivých akcií také jednoduchý kalkulátor zisku, kde může zadat počet nakoupených či prodaných akcií, poplatky za zprostředkování nákupu či prodeje a částku, za jakou byly akcie nakoupeny či prodány.

Portfolio je vlastně seznamem akcií, které chce daný obchodník sledovat. Základní zobrazení často obsahuje název firmy, obchodní zkratku, aktuální hodnotu akcie a procentuální přírůstek či úbytek v závislosti jestli šla hodnota akcie nahoru či dolů. Přidávání akcií do portfolia probíhá v aplikacích převážně na základě seznamu akcií či znalosti obchodních zkratek, popřípadě názvů.

Kromě tohoto jednoduchého výpisu základních informací nabízejí aplikace také výpisy rozšířené, ve kterých lze najít například denní minimum a maximum, což jsou nejnižší či nejvyšší ceny za jaké se akcie v ten den obchodovala. Dále poté cena při otevření burzy a taktéž cena při uzavření burzy (v případě, že je již po uzavření burzy v daný den), čas a datum obchodování a množství obchodovaných akcií. Někdy lze v rozšířených informacích nalézt také roční minimum a maximum, 52wk range (což je interval cen, jakých akcie nabývala v posledních 52 týdnech), EPS (Earning per share) a P/E. Earning per share je čistý zisk z akcie, dělený počtem kmenových akcií a indikuje ziskovost společnosti. Využívá se pro určení, zdali je daná emise podhodnocena či nadhodnocena. P/E ratio je poměr mezi tržní cenou akcie a ziskem na akcii (Price / EPS). Akcie s vysokým P/E ratio jsou považovány za rizikovější. Ukazatelů je samozřejmě daleko více, nicméně pro pochopení funkcí aplikací v této práci to plně postačí [1].

2.2 Grafy

Kromě samotného portfolia se využívají při zobrazení průběhu vývoje cen akcií na trhu také grafy. V případě webových aplikací se jedná většinou o pokročilé vykreslování grafů různých typů. Nejčastěji používanými typy grafů jsou grafy čárové, OHLC a svícnové. Tyto grafy jsou často doprovázeny sloupcovým grafem, zobrazujícím množství zobchodovaných akcií. Taktéž je v nich možno zobrazit různé typy ukazatelů či dokonce vložit ukazatele uživatelské. Mobilní aplikace naopak ve většině případů poskytují základní zobrazení čárového grafu bez možnosti úprav a bez možnosti zobrazení a zaznamenání ukazatelů.

Obr. 2.1 Čárový graf (modrá čára) spolu se sloupcovým grafem, jednoduchým klouzavým průměrem (červená čára) a exponenciálním klouzavým průměrem (zelená čára)

Nejprve se podíváme na základní typ grafu, který se většinou vyskytuje v aplikacích na všech platformách a to na čárový graf (obr. 2.1). Tento graf, na obrázku zachycen modrou čárou, ukazuje průběh vývoje zavíracích hodnot akcie během časového úseku od několika dnů až po desítky let. Na stejném obrázku pod ním se nachází sloupcový graf, zobrazující množství obchodovaných akcií v daný den (či součet množství, pokud se jedná o časovou periodu). Nakonec jsou na obrázku vidět ještě dva typy technických ukazatelů a to jednoduchý klouzavý průměr (červená čára) a exponenciální klouzavý průměr (zelená čára).

Jednoduchý klouzavý průměr (SMA) lze dostat jako průměr posledních několika zavíracích cen dle následujícího vzorce [2]:

$$SMA = \frac{(P_1 + \cdots + P_n)}{n} \tag{2.1}$$

Kde P_n je zavírací cena n-intervalu obchodních dní a n je počet dní, na jejichž základě počítáme klouzavý průměr. V případě našeho grafu je počet dní roven 20, a jak si můžete všimnout, výsledná křivka určitým způsobem kopíruje křivku čárového grafu. V praxi se používají nejčastěji SMA s periodou 5,10 a 20 dní.

Exponenciální klouzavý průměr (EMA) je již o něco složitější na výpočet. K výpočtu potřebujeme jednoduchý klouzavý průměr, multiplikátor a zavírací hodnoty.

Multiplikátor opět využívá n-interval obchodních dnů, který musí korespondovat s n-intervalem SMA (tedy pokud je v multiplikátoru použit n = 10, poté i SMA musí mít n = 10). Multiplikátor dostaneme pomocí následujícího vzorce [3]:

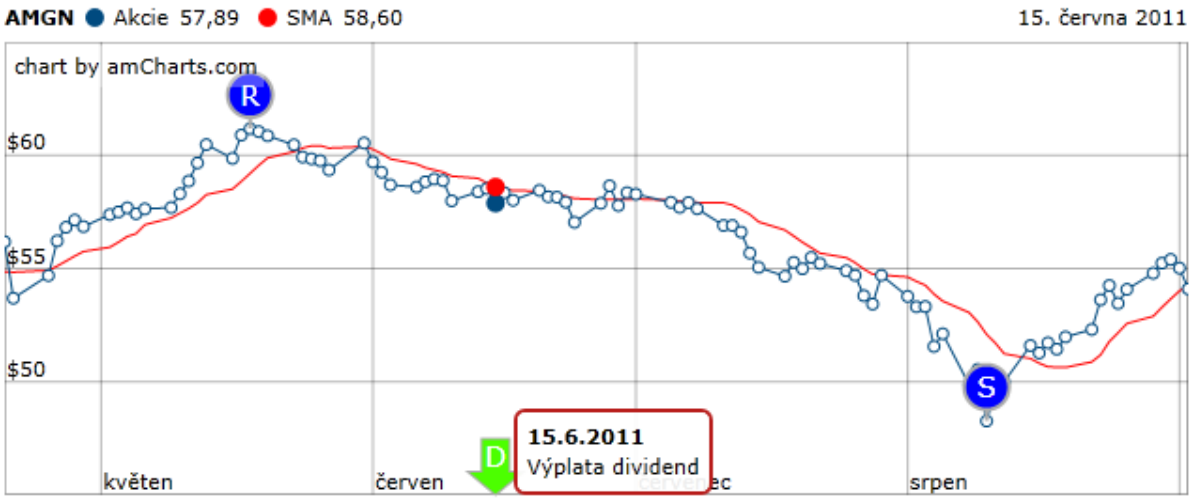
$$Multiplikátor = \frac{2}{(n+1)} \tag{2.2}$$

Jakmile tedy máme multiplikátor a SMA můžeme vypočítat EMA dle následujícího vzorce:

$$\begin{aligned} EMA(aktuální\ den) &= \big(ZavíracíCena(aktuální\ den) \\ &- EMA(předchozí\ den)\big) \times multiplikátor \\ &+ EMA(předchozí\ den) \end{aligned} \tag{2.3}$$

Kde EMA (předchozí den) je na začátku právě hodnota SMA (předchozí den). V praxi se používají nejčastěji EMA s periodou 10 a 20 dní. EMA přidává, na rozdíl od SMA, větší váhu nejbližším cenám a proto bude reagovat rychleji na aktuální změny ceny.

Klouzavé průměry jsou pouze jedním z několika využívaných ukazatelů. Těch existuje samozřejmě celá řada, ale klouzavé průměry patří mezi ty nejvyužívanější a proto jsou použity v mých aplikacích.



Obr. 2.2 Uživatelské ukazatele (R,D,S) nesoucí různé uživatelem zadané informace

Jednou z dalších možností, kterou umožňují lepší webové aplikace, je přidávání uživatelských ukazatelů a značek. Většinou se jedná o zaznamenání nákupu či prodeje akcií do grafu, výplata dividend, různé poznámky o vývoji ve společnosti a další (obr. 2.2).

Obr. 2.3 OHLC graf (nahoře) a svícnový graf (dole)

Dalším typem grafů jsou tzv. OHLC grafy a svícnové grafy (obr. 2.3). Tyto grafy zobrazují otevírací, maximální, minimální a zavírací hodnotu akcie v daném dni. V případě OHLC grafu tak, že vodorovná levá čára znamená onu otevírací hodnotu, svislá čára zobrazuje minimální a maximální hodnotu a vodorovná pravá čára zobrazuje zavírací hodnotu. Svícnový graf se liší pouze tím, že spojuje otevírací a zavírací hodnotu. Většinou pokud je tendence hodnoty akcie klesající (tzn. otevírací hodnota je větší než zavírací) vyznačuje se prvek v grafu červeně, naopak pokud je tendence hodnoty akcie rostoucí vyznačuje se v grafu zeleně.

To by bylo z této kapitoly vše. Jedná se zde pouze o nastínění teorie, základy, jež je potřeba znát pro pochopení obsahu této diplomové práce.

3 Průzkum trhu

V této kapitole si ukážeme, s jakými aplikacemi se můžeme na trhu setkat. Nejprve si představíme několik volně dostupných webových klientů pro správu portfolia a zobrazení grafů a jeden z nich si zde přiblížíme. Poté si představíme několik volně dostupných aplikací na platformě Windows Phone a také zde si jednu z nich přiblížíme.

3.1 Webový klient

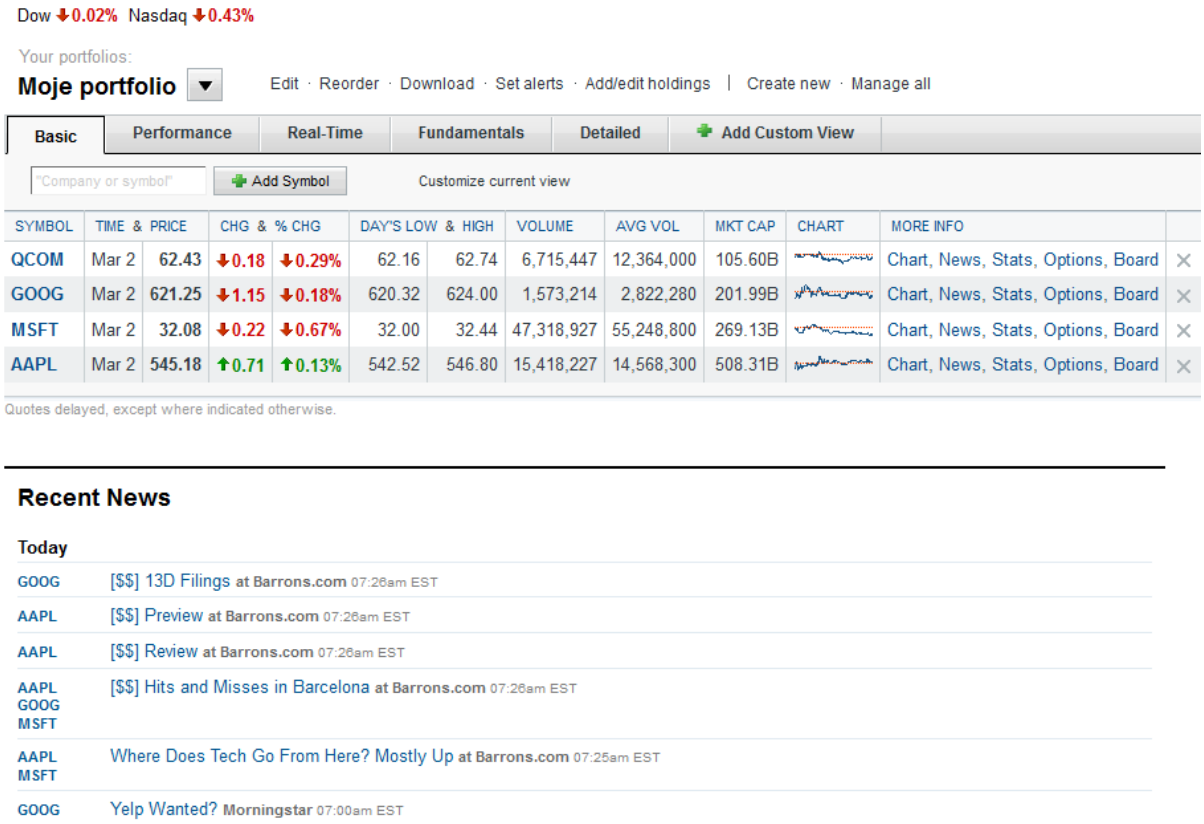
Webových klientů umožňujících správu portfolia a vykreslení nejrůznějších typů grafů a ukazatelů existuje celá řada. Můj klient se jim nesnaží nijak konkurovat (vzhledem k tomu, že používá veřejně dostupná data, která jsou většinou nabízena těmito klienty), ale měl by umožnit alespoň základní možnosti správy portfolia a zobrazení grafů, které jsou nabízeny těmito klienty.

Mezi známé webové klienty patří například Google Finance, Yahoo Finance či Finviz. Registrace a základní používání těchto služeb je většinou bezplatné, nicméně za pokročilejší funkce si již uživatel musí zaplatit. Všichni tito weboví klienti jsou napojeni na zdroje dat v reálném čase (v případě Finvizu pouze pokud si uživatel zaplatí verzi Finviz Elite). Poskytují jednoduchou správu portfolia, kdy si uživatel může vytvořit jedno i více portfolií akcií a zařadit do nich různé akcie z různých trhů. Dále je zde možnost výpočtu zisku z dané akcie a zobrazení podrobností o dané akcii. Uživatel si také může zobrazit různé typy grafů (OHLC, čárový graf, svícnový graf atd.) s možností přidání a zobrazení ukazatelů (v případě Finviz opět jen ve verzi Elite). Pro detailní přiblížení aplikace jsem vybral právě Yahoo Finance. Shrnutí funkcí bezplatných verzí aplikací poté naleznete na obrázku (Obr. 3.1).

Funkce / Aplikace	Správa portfolia	Pokročilé vykreslování grafů	Uživatelské ukazatele v grafu	Technické ukazatele	Zobrazení bez reklam	Data v reálném čase
Yahoo Finance	✓	✓	✓	✓	✗	✓
Google Finance	✓	✓	✓	✓	✗	✓
Finviz.com	✓	✗	✗	✗	✗	✗
Moje aplikace	✓	✓	✓	✓	✓	?

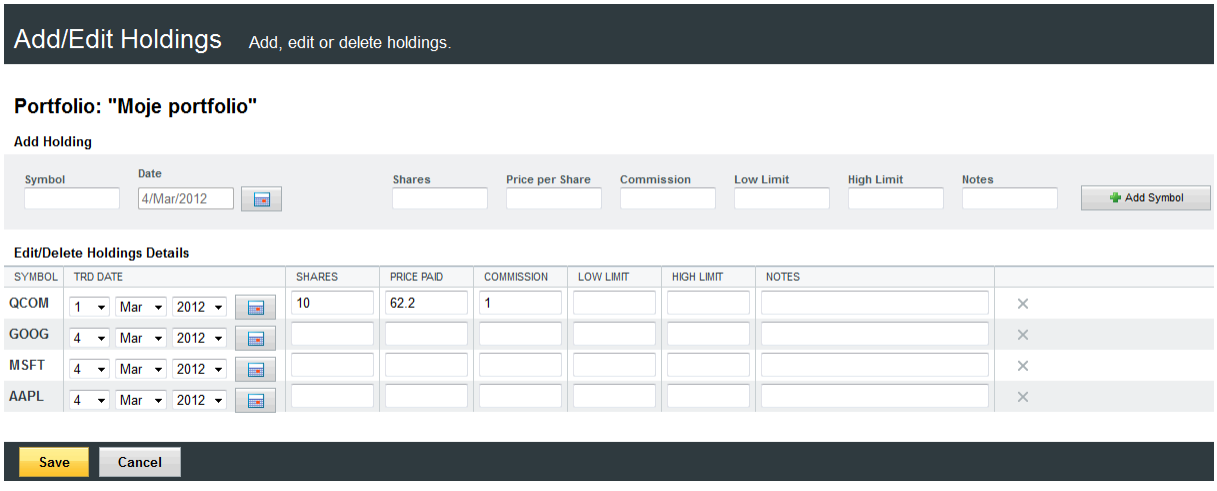
? Záleží na zdroji dat, které využívá aplikace pro plnění databáze daty z akciových trhů

Obr. 3.1 Obrázek shrnující funkce bezplatných verzí webových aplikací



Obr. 3.2 Moje portfolio s několika akciemi

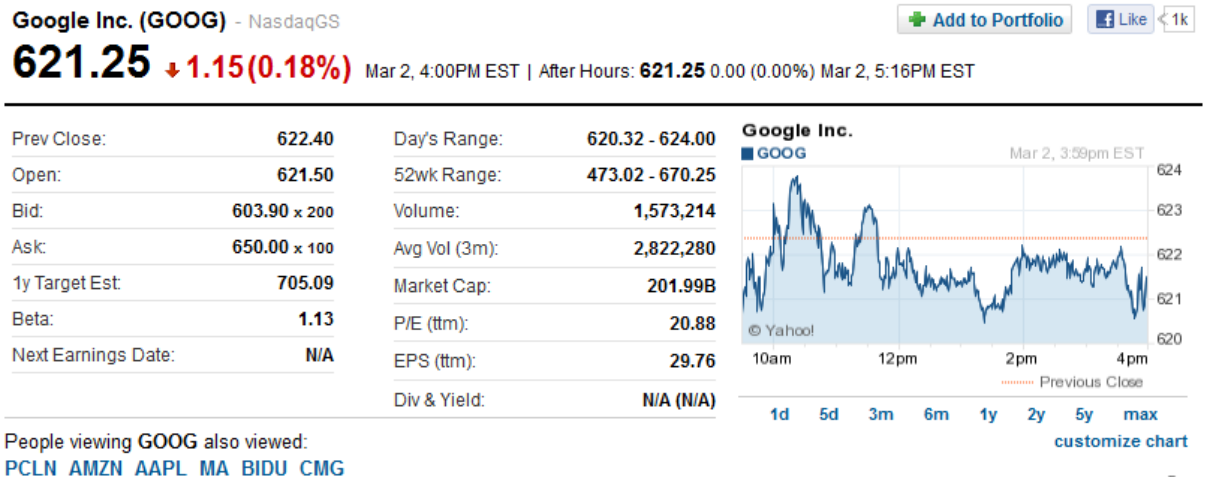
Na obrázku (Obr. 3.2) je vyobrazeno portfolio, které jsem si vytvořil na Yahoo Finance. Obsahuje základní informace o čtyřech akciích (QCOM, GOOG, MSFT, AAPL). Dále si můžete všimnout, že jsou zde pod portfoliem zobrazeny aktuality o daných firmách. Pokud bychom chtěli přidat informace o námi nakoupených či prodaných akciích, taktéž to zde není problémem.



Obr. 3.3 Přidání informací o nákupu akcií

Po kliknutí na tlačítko Add/Edit holdings na obrázku (Obr. 3.2) se dostaneme na obrazovku (Obr. 3.3), na které můžeme editovat, smazat či přidat počet nakoupených kusů, včetně ceny za nákup jedné akcie a poplatků. Je zde také možnost nastavit alarm v případě, že akcie překročí určitou nejnižší

či nejvyšší hodnotu. Pokud klikneme na stránce s akciemi (Obr. 3.2) na jednu ze zkratek akcií, zobrazí se nám detailní informace o akcii (Obr. 3.4).



Obr. 3.4 Upřesňující informace o akcii

Chceme-li vykreslit grafy akcie na obrázku (Obr. 3.4) klikneme na zobrazený jednoduchý čárový graf a to nás přenese do detailnějšího zpracování grafů (Obr. 3.5). Zde je zachycen jeden z mnoha nabízených typů grafů, které si zde uživatel může zobrazit společně s mnoha typy různých ukazatelů (na obrázku si můžete všimnout ukazatele SMA).

Obr. 3.5 Jeden z mnoha dostupných grafů na Yahoo Finance

Samozřejmě zde představené věci představují zlomek schopností toho, co Yahoo Finance umí. Nicméně stejně jako u Google Finance je bezplatnost vykoupena zobrazováním reklam, což nemusí všem zrovna vyhovovat. Většinu zde představených věcí by měl umět i můj výsledný produkt.

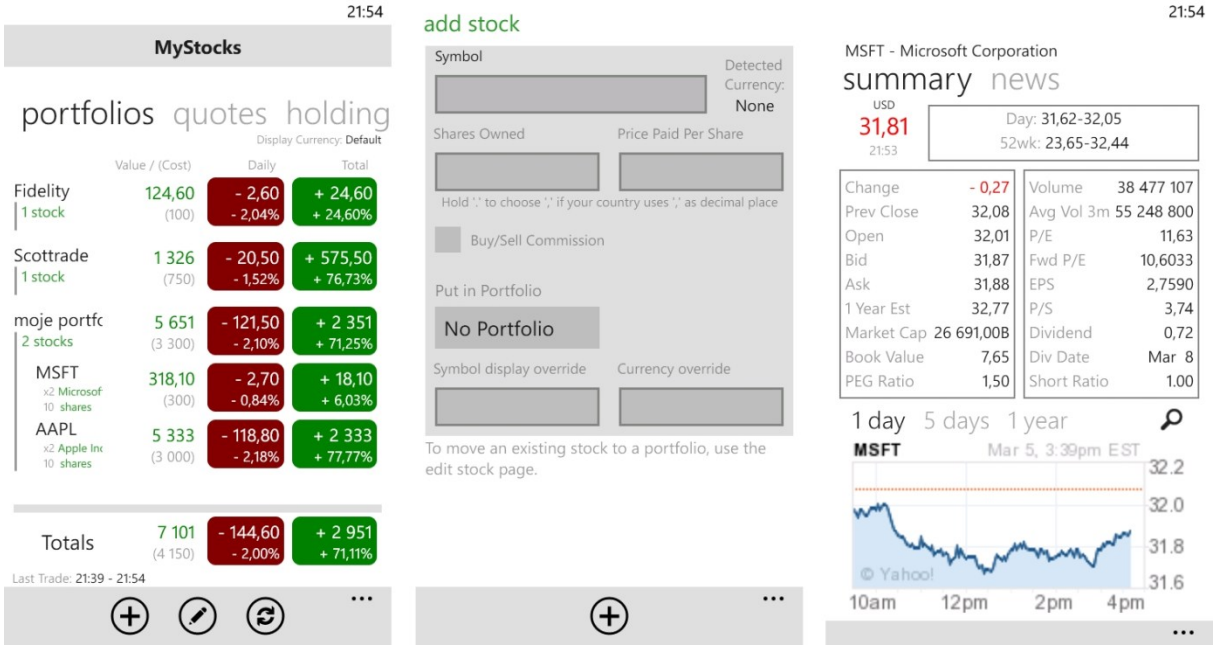
3.2 Mobilní klient

Platforma Windows Phone je relativně mladá a tudíž počet volně dostupných aplikací existujících na tuto platformu není vůbec velký. Výhodou je, že některé volně dostupné aplikace se liší od svých placených verzí pouze přítomností zobrazování reklamy. Je nutno dodat, že některé aplikace nemusí být ve všech Marketplace (místo pro nákup a stahování aplikací pro platformu Windows Phone) dostupné. Vycházel jsem tedy z aplikací dostupných na českém Marketplace v době psaní tohoto textu. Mezi nejpopulárnější aplikace zdarma dle žebříčku patří MSN Money Stocks, MyStocks Portfolio a Watchlist. Všechny tyto aplikace umožňují vytvoření vlastního portfolia a zobrazení podrobnějších informací o akciích s možností vykreslení opravdu jednoduchého čárového grafu (Watchlist jako jediný nabízí možnost pokročilejšího vykreslení čárového grafu s možností pohybu po grafu a zobrazení hodnot po kliknutí). Možnost výpočtu zisku z nakoupených akcií poskytuje ovšem pouze aplikace MyStocks Portfolio, kterou si zde trochu podrobněji přiblížíme. Shrnutí funkcí bezplatných verzí aplikací poté naleznete na obrázku (Obr. 3.6).

Aplikace \ Funkce	Správa portfolia	Pokročilé vykreslování grafů	Uživatelské ukazatele v grafu	Technické ukazatele	Zobrazení bez reklam	Data v reálném čase
MSN Money Stocks	✓	✗	✗	✗	✓	✗
MyStocks Portfolio	✓	✗	✗	✗	✗	✗
Watchlist	✓	✓	✗	✗	✓	✗
Moje aplikace	✓	✓	✓	✓	✓	?

? Záleží na zdroji dat, které využívá aplikace pro plnění databáze daty z akciových trhů

Obr. 3.6 Shrnutí funkcí bezplatných verzí mobilních aplikací



Obr. 3.7 Zleva – Portfolia, přidání akcie a zobrazení detailních informací s jednoduchým grafem

Na obrázku (Obr. 3.7) je vlevo zobrazena správa portfolia, kterých v této aplikaci může býti i více. U každého portfolia je navíc uveden zisk či ztráta z nakoupených akcií. Uprostřed je poté přidání samotné akcie včetně počtu vlastněných akcií a ceny za jednu akcii do portfolia. Pokud klikneme na danou akcii v portfoliu, zobrazí se nám její detailní přehled, včetně jednoduchého čárového grafu což je zobrazeno na obrázku úplně vpravo. Ukazatele zde bohužel není možno zaznamenat.

Závěrem bych chtěl dodat, že množství aplikací pro platformu Windows Phone 7 stále roste, tudíž, co bylo aktuální v době psaní tohoto textu, nemusí být aktuální v době, kdy toto bude čtenář číst. Všechny výše zmíněné funkce by měly být obsaženy také v mé mobilní aplikaci, ovšem včetně pokročilejšího vykreslování grafů.

4 Využité technologie

Vzhledem k tomu, že aplikace jsou celkem tři a každá funguje v jiném prostředí, bylo nutno vybrat technologie, na kterých budou tyto aplikace postaveny. Největší nabyté zkušenosti mám s jazykem C# a tudíž volba padla na technologie firmy Microsoft.

Pro aplikaci běžící na serveru a pro webového klienta jsem využil technologii .NET ve verzi 4.0. Pro vykreslování grafů v prostředí webu je využita technologie Silverlight v aktuální verzi 5.0. Nakonec pro mobilní aplikaci byla zvolena platforma Windows Phone 7 respektive její nejaktuálnější verze Windows Phone 7.5 též známá jako Mango. Všechny tyto technologie umožňují programování v jazyce C# a nyní se zde na ně podíváme trochu podrobněji.

4.1 Technologie .NET v kostce

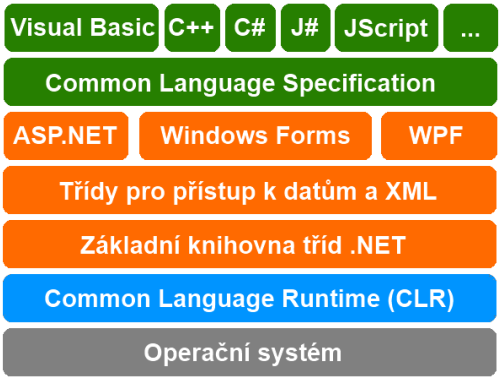
Technologie .NET byla vyvinuta firmou Microsoft a světlo světa spatřila v roce 2002 ve své první verzi 1.0 společně s novým moderním objektově orientovaným jazykem C#. Od Windows verze Server 2003 je také přímo součástí systému. Na ostatní systémy Windows ji lze doinstalovat. Systémy Windows 98, ME a 2000 podporují nejvýše verzi 2.0. Od systému Windows XP je to poté až nejnovější verze 4.0. Ve vývoji je nová verze 4.5, která přijde s nadcházejícím operačním systémem Windows 8.

Mezi hlavní výhody této technologie patří:

- Objektově orientované programování
- Jazyková nezávislost
- Lepší podpora dynamických webových stránek
- Účinný přístup k datům
- Sdílení kódu
- Zlepšení zabezpečení
- Instalace s nulovým dopadem
- Podpora webových služeb
- Jazyk C#

Více informací o jednotlivých bodech naleznete v této knize [4].

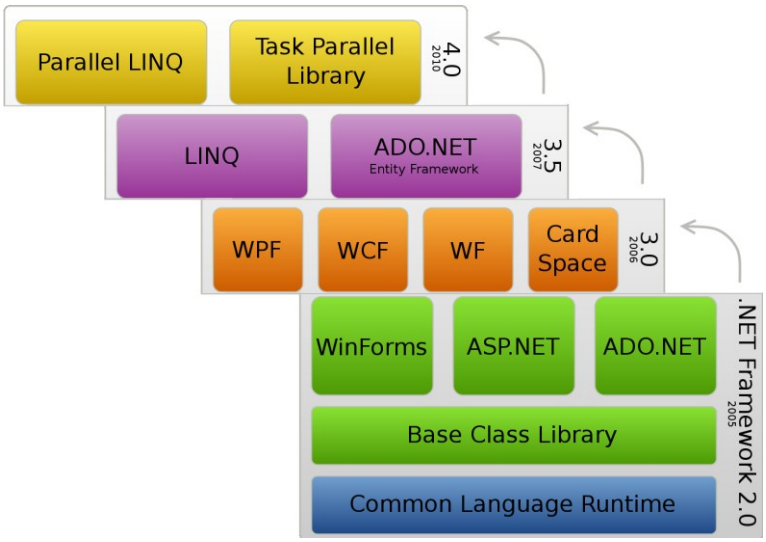
.NET framework má dvě hlavní komponenty, kterými jsou Common Language Runtime (CLR) a knihovna tříd .NET Framework. Na obrázku (Obr. 4.1) si můžete prohlédnout základní schéma .NET Frameworku.



Obr. 4.1 Základní schéma .NET Frameworku

CLR je běhové prostředí, které zajišťuje běh a kompilaci programů. Toto prostředí spravuje kód v průběhu provádění, poskytuje základní služby, jako jsou správa paměti, správa vláken, vzdálená komunikace, obsluha výjimek a taktéž se stará o dodržování typové bezpečnosti a dalších forem přesnosti kódu, které zajištují jeho bezpečnost a robustnost. Tomuto kódu se říká řízený (managed) kód. Kompilace poté probíhá ve dvou krocích. V prvním kroku se kód zkompiluje do tzv. MSIL (Microsoft Intermediate Language). Tento kód je podobný strojovému kódu, ale jeho výhodou je, že je nezávislý na platformě. Všude tam, kde je nainstalován .NET Framework, bude tento kód spustitelný. Poté ve druhém kroku CLR zkompiluje MSIL do strojového kódu dané platformy. Neprovádí se ovšem kompletní kompilace, ale pouze částečná kompilace toho, co je pro běh aplikace zatím potřebné. V případě volání metody, která ještě nebyla zkompilována ji JIT kompilátor (kompilátor překládající MSIL kód za běhu – Just-in-time) přeloží a výsledek si ponechá pro další volání. Není tedy potřeba překládat vše a tudíž je spuštění aplikace velice rychlé.

Knihovna tříd .NET Frameworku je komplexní, objektově orientovaná kolekce znovupoužitelných typů, kterou lze využít pro vývoj tradičních aplikací v příkazové řádce, pro vývoj aplikací s grafickým uživatelským rozhraním (GUI) až po aplikace založené na technologii ASP.NET (o té později blíže), například pro tvorbu webových formulářů či webových služeb. Samozřejmě možností využití je ještě daleko více a s každou novou verzí .NET Frameworku přibývají i nové funkce. Přehled vývoje .NET Frameworku včetně funkcí, které přibyly v jednotlivých verzích si můžete prohlédnout na následujícím obrázku (Obr. 4.2).



Obr. 4.2 Vývoj .NET Frameworku [5]

Kromě popsaných výhod má .NET Framework samozřejmě také své nevýhody. Vzhledem k tomu, že je zde nějaká mezivrstva (tou je právě ono zmiňované CLR), která se stará o běh, jsou aplikace většinou paměťově náročnější a oproti nativním aplikacím mohou být pomalejší v řádech procent, ale i více. Ačkoliv lze z .NET Frameworku volat neřízený (unmanaged) kód pomocí wrapperů, existují aplikace, kde toto volání není tak efektivní jako použití neřízeného kódu přímo (komplexní počítačové hry). Toto ovšem není náš případ.

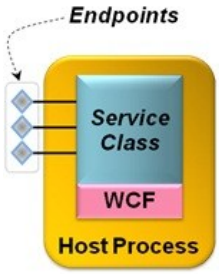
V této kapitole jsem zjednodušeně přiblížil technologii .NET, pro další informace můžete navštívit zdroje ze kterých jsem čerpal [6, 7]. Mezi klíčové součásti .NET Frameworku, které jsou využity v mých aplikacích, patří WinForms, WCF, ASP.NET a Task Parallel Library. Nyní si je trochu přiblížíme.

4.1.1 Windows Forms

Windows Forms je knihovna pro vytváření tzv. chytrých klientů. Chytří klienti jsou graficky bohaté aplikace, které lze snadno nasadit, aktualizovat a které mohou fungovat jak v dostupnosti internetového připojení tak i bez něj. Na rozdíl od Windows API jsou aplikace vytvořené v tomto grafickém API bezpečnější a robustnější, neboť se jedná opět o řízený kód. Windows API je zde vlastně voláno pomocí wrapperů z řízeného kódu. Pomocí Windows Forms lze vytvářet aplikace, které jsou široce zaměřené, umožňující běžnou funkcionalitu jako například práci se soubory, komunikaci se vzdálenými PC po síti, zobrazování nejrůznějších typů informací apod. To vše v uživatelsky přívětivém prostředí. Hlavním stavebním prvkem Windows Forms je formulář (form) – prázdné okno, do kterého vývojář vkládá ovládací prvky (controls) – například tlačítka, textová pole apod. a vytváří odezvu na uživatelské akce provedené na těchto prvcích. Jedná se tedy o technologii založenou na událostech. Jakmile uživatel klikne např. na tlačítko, dojde k vygenerování události, aplikace zareaguje na tuto událost a zpracuje ji [8].

4.1.2 WCF

WCF tedy Windows Communication Foundation je framework pro vytváření servisně orientovaných aplikací (SOA). WCF sjednocuje technologie používané pro distribuované aplikace. Kombinuje funkčnost webových služeb ASP.NET (které poskytují interoperabilitu), .NET Remoting (výkonnost), Message Queuing (spolehlivost) a Enterprise Services (distribuované transakce a správa životního cyklu objektů). Díky tomu se zjednodušuje vývoj, údržba i nasazení distribuovaných aplikací. Základním stavebním kamenem WCF aplikací jsou služby (services). Servisní třída (service class) implementuje jednu nebo více metod. Služba samotná běží v hostitelském procesu (host proces), neboť se většinou jedná o knihovnu, která potřebuje hostitelských proces, ve kterém by mohla být spuštěna. Služby si můžeme představit jako systém, se kterým komunikujeme pomocí koncových bodů (endpoints). Koncový bod slouží k přijímání zpráv a odesílání odpovědí. Služba může mít jeden ale i více koncových bodů (Obr. 4.3) [9].



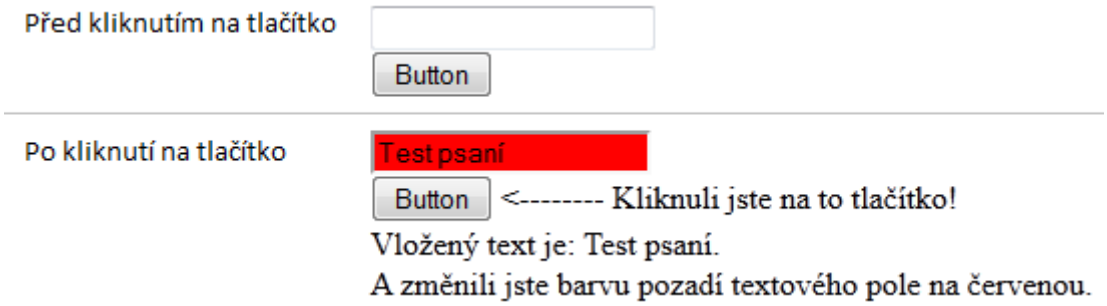
Obr. 4.3 WCF služba běžící v hostitelském procesu a vystavující jeden nebo více koncových bodů [10]

4.1.3 ASP.NET

Technologie ASP.NET je souborem funkcí a objektů, které umožňují vytvářet webové stránky pracující na serveru. ASP.NET je stejně jako .NET nezávislé na programovacím jazyku (jedinou podmínkou je, aby existoval kompilátor, který umožní zdrojový kód přeložit do jazyka MSIL). ASP.NET využívá všech výhod CLR a využité programovací jazyky patří tedy na rozdíl od většiny skriptovacích jazyků, které jsou interpretované, mezi jazyky kompilované. Chod webových aplikací je v ASP.NET řízen, stejně jako u Windows Forms, událostmi, díky čemuž lze vytvářet vysoce interaktivní stránky. Výhodou ASP.NET je také velké množství webových ovládacích prvků, kdy je možno vytvářet také prvky vlastní a tudíž lze dosáhnout vysoce propracovaného uživatelského rozhraní. Další nespornou výhodou je, že ASP.NET dokáže z požadavku klienta zjistit, ve kterém prohlížeči bude stránka zobrazena a podle toho sestavit její zdrojový kód tak, aby se stránka zobrazila korektně. Řeší se tedy problémy s nekompatibilními prohlížeči. Kromě výhod existují samozřejmě také nevýhody. Největší nevýhodou je, že pro provozování web aplikací je nutno mít nainstalovaný IIS (Internet Information Services), což je webový server s kolekcí rozšiřujících modulů, vytvořený Microsoftem pro operační systém Windows. Tento server sice lze doinstalovat do klasických OS Windows, nicméně budete, dle verze, limitováni maximálním počtem současných připojení. Abyste tuto limitaci odstranili, musíte využít serverovou verzi Windows, což se může prodražit.

Nyní si názorně vysvětlíme samotný princip ASP.NET. Klient bude chtít přistoupit na stránku, zadá tedy do prohlížeče požadovanou adresu a potvrdí. Požadavek se odešle na server, zde se stránka

vyhledá a CLR přeloží a spustí její programový kód. Z jeho výsledků server sestaví HTML kód výsledné stránky a ten odešle zpět prohlížeči. Ten poté zobrazí klientem zvolenou stránku. Jedná se tedy o standardní princip klient/server. ASP.NET jde ovšem v tomto dále, a jak již bylo řečeno v předchozím odstavci, zavádí událostmi řízený model. Pokud tedy dojde k nějaké události, například uživatel stiskne na stránce tlačítko nebo změní výběr položky apod., odešle webový prohlížeč informaci o této události na server, kde je možno na tuto událost zareagovat například změnou obsahu stránky. Jakmile je stránka změněna, pošle se zpět webovému prohlížeči a ten ji zobrazí (Obr. 4.4). Aby byla zaručena interaktivita stránek, využívá server skriptů v jazyce JavaScript, které vkládá při sestavování do výsledného HTML kódu [11].



Obr. 4.4 Ukázka jednoduché ASP.NET stránky před a po kliknutí na tlačítko

4.1.4 Task Parallel Library (TPL)

Je soubor veřejných typů a API, který se poprvé objevil v .NET Frameworku verze 4. Cílem TPL je poskytnout vývojářům větší produktivitu při vývoji aplikací přidáním tříd, které umožňují práci s více vlákny a paralelismus. TPL se stará o dynamické rozložení zátěže tak, aby co nejlépe využil všechna jádra procesoru. TPL se dále stará o rozdělování práce vláknům, plánování, podporu zrušení, management stavů atd. Použitím TPL lze maximalizovat výkon kódu, nicméně to přináší také negativa v podobě zvýšení komplexnosti aplikace a nové problémy spojené s více vlákny a využitím více procesorových jader. Více informací naleznete v tomto zdroji [12].

Tímto jsem shrnul základní informace o jedné z použitých technologií. Přiblížili jsme si Windows Forms spolu s Task Parallel Library, které využívá moje aplikace běžící na serveru. Dále jsme si něco řekli o ASP.NET, jež je klíčovou technologií mého webového klienta. No a také jsem se zmínil o WCF, které v případě mé aplikace využívá pro hostování ASP.NET, a poskytuje datové služby pro grafy, které jsou součástí webového klienta a také pro mobilního klienta.

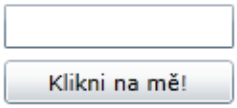
4.2 Silverlight

Mezi další technologie firmy Microsoft patří Silverlight. Cílem této technologie byl vývoj bohatých internetových aplikací (tedy aplikací, které by přinesly funkcionalitu klasických stolních aplikací do prostředí webu), které by byly nezávislé jak na použité platformě, tak na použitém webovém prohlížeči. Samotný Silverlight je klientským běhovým prostředím. První verze byla vydána v roce 2007, nicméně ta neměla s pozdějšími verzemi nic moc společného, tedy až na jazyk XAML (který vychází z XML) a slouží ke tvorbě grafického uživatelského rozhraní. Až verze 2.0 přinesla

revoluci. Tato verze byla první, která byla postavena na podmnožině .NET Frameworku (vypustili funkce nepotřebné pro vývoj bohatých internetových aplikací). Poslední verze 5.0, která byla vydána na konci roku 2011, bude patrně poslední verzí vůbec, neboť ve webovém prostředí se bude postupně přecházet na technologii HTML 5. Silverlight tímto nicméně nevymizí, neboť se stal základním vývojářským prostředím na platformě Windows Phone (o té později).

Výhody využití podmnožiny .NET Frameworku jsou jasné. Vývojář si může vybrat z široké palety nabízených programovacích jazyků. Aplikaci jsou spustitelné všude tam, kde je nainstalován Silverlight (64bitová verze pro Windows zabírá pouhých 11,6MB). Aplikace postavené na technologii Silverlight mohou být spouštěny jak ve webovém prohlížeči tak i mimo něj. Jedná se o řízený kód, takže aplikace jsou bezpečnější a robustnější. Celý Silverlight obsahuje velké množství funkcí jako například přehrávání nejrůznější typů video formátů a zvukových formátů. Umí využít grafický procesor klientského počítače pro akceleraci přehrávání videa, ale také 2D či 3D obsahu. Dále obsahuje velké množství ovládacích prvků, včetně možnosti definovat prvky vlastní. Podporuje práci s obrázky a různá vstupní zařízení (webovou kameru, klávesnici, myš atd.). Silverlight toho umí samozřejmě mnohem více, nicméně na to zde již není místo, proto pokud chcete více informací o této technologii, doporučuji knihu, ze které jsem čerpal, a která je uvedena na konci této kapitoly.

Nicméně opět jako každá jiná technologie, i Silverlight má své slabiny. Verze pro operační systém Linux neexistuje vůbec (ačkoliv je ve vývoji svobodná verze Silverlightu s názvem Moonlight, na jejímž vývoji se podílí firma Novell, s kompatibilitou to není úplně ideální, neboť vývoj oproti aktuálním verzím Silverlightu zaostává). Taktéž nejsou podporovány všechny webové prohlížeče, takže spuštění Silverlightu jako součásti webové stránky může býti problém.




```
<UserControl x:Class="SilverlightApplication1.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="200" d:DesignWidth="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <Button Content="Klikni na mě!" Height="23" HorizontalAlignment="Left" Margin="81,41,0,0"
                Name="button1" VerticalAlignment="Top" Width="120" Click="button1_Click" />
        <TextBox Height="23" HorizontalAlignment="Left" Margin="81,12,0,0" Name="textBox1"
                 VerticalAlignment="Top" Width="120" />
    </Grid>
</UserControl>
```

Obr. 4.5 Nahoře jednoduché textové pole a tlačítko, dole poté jejich definice v jazyce XAML

Technologie Silverlightu je stejně, jako tomu bylo u ASP.NET či WindowsForms, založena na událostech, takže zde tento princip nebudu blíže popisovat. Vývojář vytvoří uživatelské rozhraní pomocí jazyka XAML. K tomuto souboru XAML existuje tzv. code-behind soubor, který je napsán v jednom z podporovaných jazyků a který se stará o funkcionalitu. Co si zde ale ukážeme je právě jazyk XAML (Obr. 4.5). Tento jazyk, jak již bylo napsáno výše, vychází z jazyka XML. Vývojář definuje ovládací prvek pomocí značek (tags) podobně jako je tomu u HTML. Silverlight obsahuje

velké množství nejrůznějších ovládacích prvků, navíc umožňuje vývojáři definovat vlastní. To přináší řadu výhod, mezi které patří například možnost ovládací prvek znovu využít a taktéž, v případě komplexního uživatelského rozhraní, zvýšit přehlednost kódu.

To by bylo ve zkratce k této technologii vše. V mé webové aplikaci je tato technologie využita pro zobrazení grafů, které jsou díky tomu velice interaktivní. Pokud vás zajímají další podrobnosti o této technologii, doporučuji využít knihy, ze které jsem čerpal [13].

4.3 Windows Phone

Windows Phone je relativně mladá platforma, která přišla na trh koncem roku 2010. Microsoft kompletně zahodil předchozí Windows Mobile a vytvořil zbrusu novou platformu postavenou na technologii Silverlight a XNA. Původní Windows Phone 7 vycházel z technologie Silverlight 3, kterou rozšířil o funkce pro mobilní zařízení. Aktuální verze Windows Phone 7.5 poté přidala velké množství nových funkcí a vychází ze Silverlightu 4. Samotné XNA je multiplatformní technologie pro vývoj 2D i 3D aplikací, tedy převážně her. XNA stejně jako Silverlight běží na podmnožině .NET Frameworku, což sebou nese již u Silverlightu zmíněné výhody. Vývojář si může vybrat, kterou z oněch dvou technologií použije pro vývoj, nelze je však zatím kombinovat. Microsoft rovněž stanovil přísné hardwarové požadavky, které musí výrobci přístrojů s tímto systémem dodržovat. Aktuálně byly mírně přehodnoceny (z těch hlavních je to 800MHz procesor ARM v7 architektury, 256MB RAM, 8GB ROM, kapacitní displej s rozlišením 800x480). Aby Microsoft nalákal vývojáře, vydal vývojové nástroje pro tuto platformu zdarma ke stažení. Taktéž je k dispozici kniha, zabývající se vývojem pro tuto platformu a to zcela zdarma [14].

Jako každá nová technologie, také Windows Phone má své nevýhody. Tou největší je nemožnost programování v nativním kódu, což zamezuje vývojářům převézt některé aplikace na tuto platformu bez nutnosti přepsání aplikace či dokonce úplně. To by se ovšem mělo změnit s nadcházejícím systémem Windows Phone 8. Mezi další nevýhody patří relativní uzavřenost tohoto systému, která znemožňuje například volně stáhnout aplikaci z internetu, využívat telefon jako přenosný disk apod. Uživatelé mohou aplikace do tohoto telefonu koupit výhradně přes Windows Marketplace (což je obchod speciálně určený pro tuto platformu). Je zde sice možnost odemknutí telefonu, ale ta je hlavně pro vývojáře a za roční poplatek. Možnost odemknutí telefonu pro spotřebitele byla zrušena těsně před dokončením této práce.

Mobilní aplikace, kterou jsem vyvinul, je určena právě pro tento mobilní operační systém a všechny telefony s tímto systémem ve verzi 7.5 by měly být schopny jejího provozu.

4.4 MS SQL Server

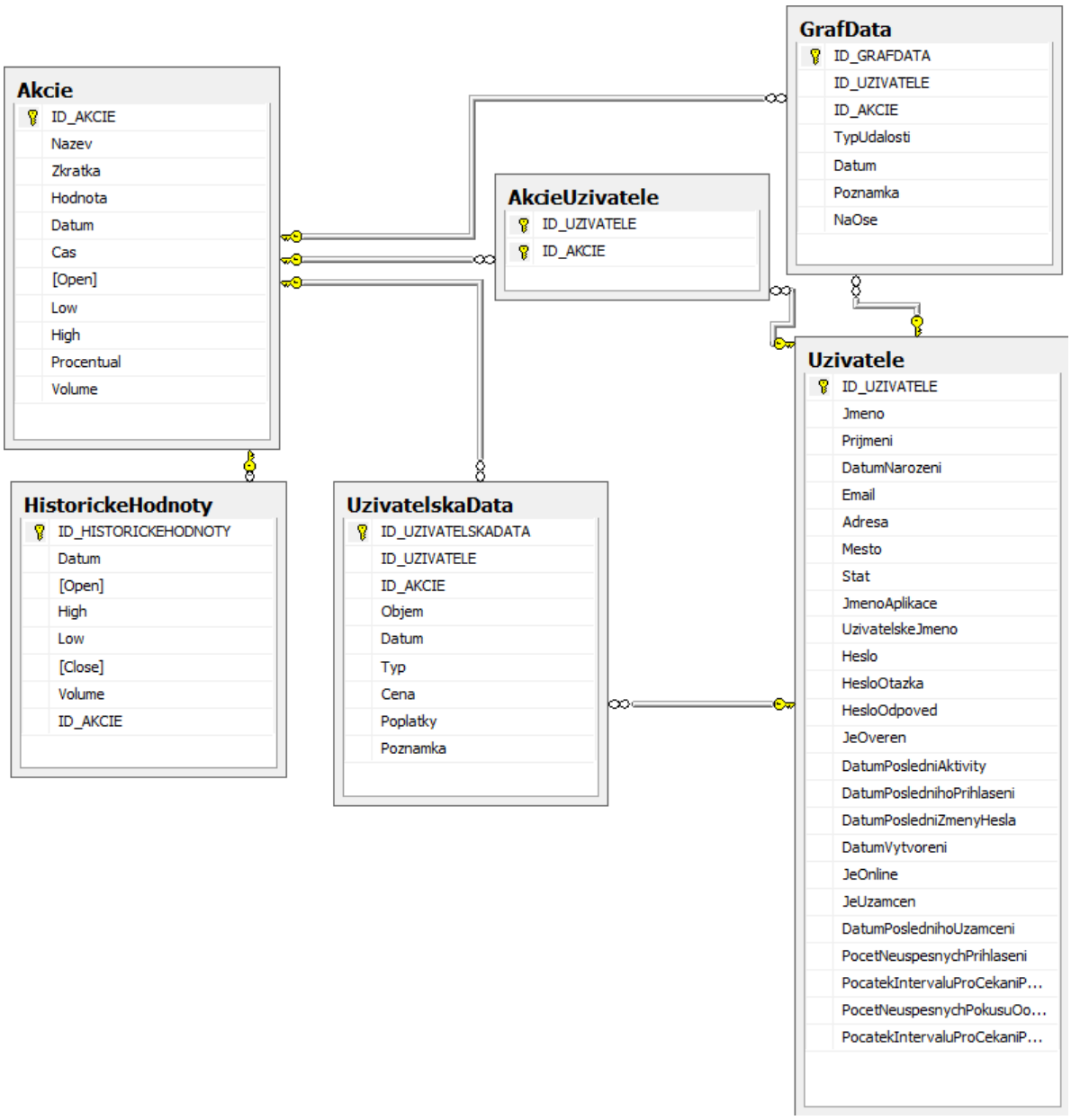
MS SQL Server je jedním z mnoha relačních databázových serverů. Tento databázový systém poskytuje řadu efektivních nástrojů pro správu a analýzu dat. Důraz je taktéž kladen na bezpečnost uložených dat, lepší škálovatelnost výkonu, dostupnost a co nejjednodušší správu. Pro práci s databází využívá technologii Transact-SQL, což je rozšíření standardního SQL jazyka firmou Microsoft a Sybase.

Tato technologie je použita pro práci s daty ve všech mých aplikacích. Pro bližší informace o MS SQL Serveru doporučuji tuto knihu, ve které je tato technologie dopodrobna popsána [15].

Tímto jsem zde shrnul hlavní použité technologie, na kterých jsou postaveny moje aplikace. Další technologie, které jsou součástí aplikací, a které mají co dočinění s funkcionalitou některých částí jednotlivých aplikací, budou zmíněny při vlastním popisu implementace.

5 Návrh aplikací

Tato část práce má za úkol přiblížit návrh jednotlivých aplikací. Všechny aplikace využívají pro práci s daty databáze MS SQL, jak zde již bylo zmíněno. Na následujícím obrázku si můžete prohlédnout schéma databáze (Obr. 5.1). Je to poměrně jednoduchá databáze, nicméně obsahuje vše potřebné pro práci s daty pro všechny tři aplikace. Samotná databáze taktéž obsahuje 42 procedur pro zjednodušení práce s daty. Procedury se starají o vkládání, mazaní, editaci apod. v rámci všech tabulek. Nepoužil jsem zde kvůli přehlednosti a relativně malé složitosti databáze žádné objektově relační mapování.



Obr. 5.1 Schéma databáze

Serverová aplikace využívá pouze tabulky Akcie a HistorickeHodnoty. Klientské aplikace poté přistupují ke všem tabulkám. Pro práci s databází mimo mé aplikace jsem využil SQL Server Management Studio, které je součástí instalace MS SQL Serveru. Všechny aplikace byly vytvořeny za pomocí vývojového prostředí Microsoft Visual Studio 2010. Pro více informací o tomto vývojovém prostředí doporučuji tuto publikaci [16].

V následujících kapitolách si shrneme hlavní funkční a nefunkční požadavky, které byly kladeny na výsledné aplikace. Taktéž si ukážeme diagramy hlavních tříd pro všechny aplikace. Tyto diagramy byly vygenerovány pomocí vývojového prostředí Visual Studio 2010. Jednotlivé diagramy jsem poté upravil tak, abych rozdělil jednotlivé třídy dle využití.

5.1 Návrh serverové aplikace AkcieDataDownloader

Aplikace byla navržena s cílem chodu na serveru. V ideálním případě by měl být server vybaven procesorem s alespoň dvěma jádry a dostatkem operační paměti. Návrh aplikace počítá s optimalizací využití paměti a aplikace by dlouhodobě neměla využívat více jak 100MB operační paměti.

5.1.1 Funkční požadavky

Tabulka 5.1 Hlavní funkční požadavky kladené na aplikaci AkcieDataDownloader

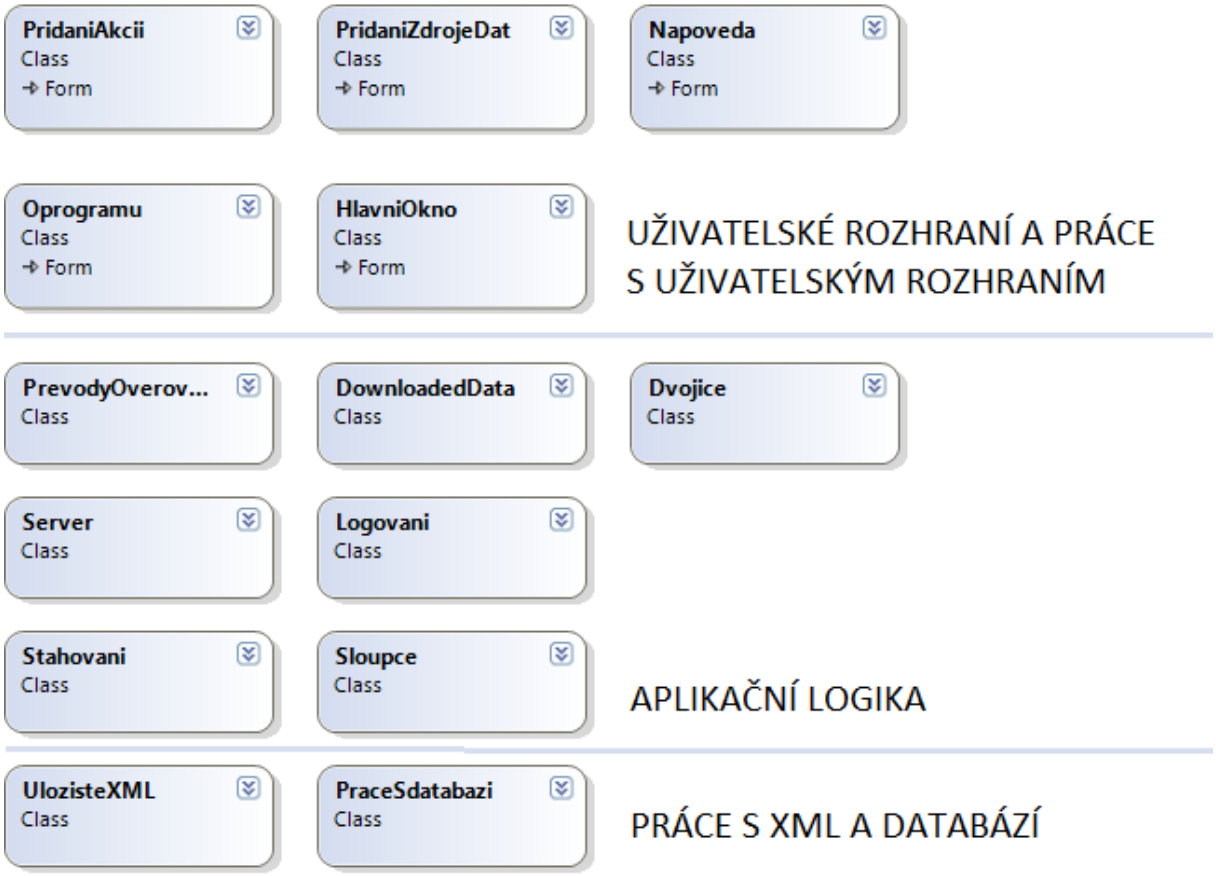
Značení požadavku	*Popis*
PK001	*Aplikace umožní definovat a poté zvolit MS SQL server pro ukládání dat*
PK002	*Aplikace bude po úspěšném připojení na server automaticky vytvářet databázi*
PK003	*Aplikace umožní definovat zdroje pro historická data i pro data v reálném čase*
PK004	*Aplikace umožní nastavit intervaly aktualizací pro historická data i data v reálném čase*
PK005	*Aplikace umožní přidat, smazat či editovat akcie uložené v databázi*
PK006	*Data historických akcií a data akcií v reálném čase budou stahována nezávisle na sobě*
PK007	*Aplikace si bude ukládat informace o chybách při stahování a zpracování akciových dat*
PK008	*Nastavení aplikace budou ukládána do souborů XML*
PK009	*Aplikace bude mít možnost zobrazení jednoduché nápovědy*
PK010	*Aplikace bude podávat v průběhu práce informace o aktuálně vykonávané činnosti na obrazovku*

5.1.2 Nefunkční požadavky

Tabulka 5.2 Hlavní nefunkční požadavky kladené na aplikaci AkcieDataDownloader

Značení požadavku	*Popis*
PK001	*Aplikace bude mít vygenerovánu dokumentaci a kód aplikace bude okomentován*
PK002	*Uživatelské rozhraní bude v českém jazyce*
PK003	*Aplikace bude vyžadovat .NET Framework 4.0 a vyšší*

5.1.3 Třídní diagram



Obr. 5.2 Hlavní třídy aplikace AkcieDataDownloader

5.2 Návrh klientské aplikace AkcieWebKlient

Tato webová aplikace by měla být hostována na stejném serveru jako aplikace AkcieDataDownloader (nicméně není to povinností). Na serveru musí být zajištěna dostatečná konektivita s možností rozšíření do budoucna v případě většího počtu přistupujících klientů.

5.2.1 Funkční požadavky

Tabulka 5.3 Hlavní funkční požadavky kladené na aplikaci AkcieWebKlient

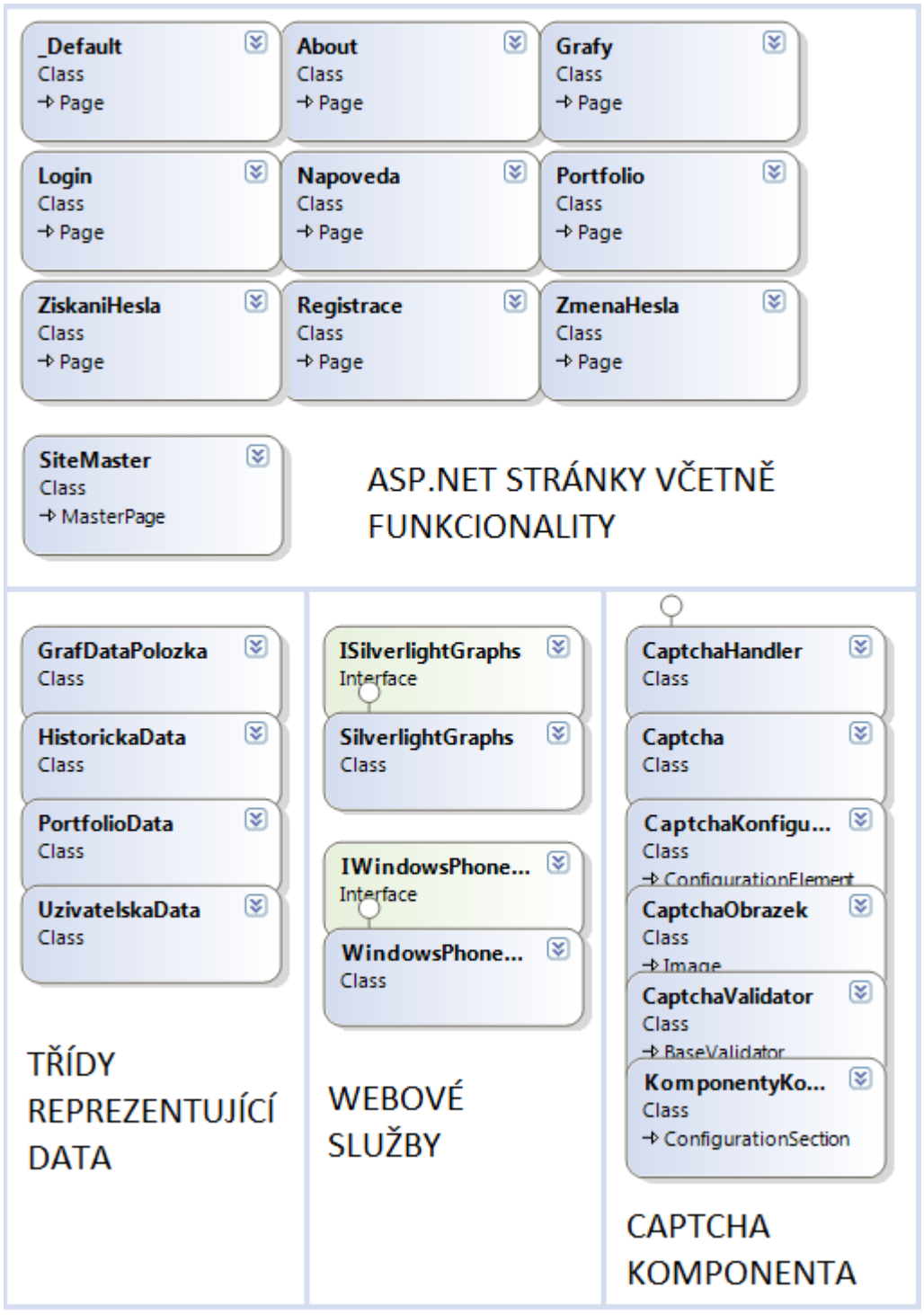
Značení požadavku	*Popis*
PK001	*Aplikace umožní registraci a přihlašování uživatelů*
PK002	*Aplikace umožní získání zapomenutého hesla a možnost změny hesla uživatelů*
PK003	*Aplikace bude podporovat jednoduchou správu portfolia akcií*
PK004	*Aplikace umožní uživatelům vést informace o nákupech a prodejích akcií*
PK005	*Aplikace bude umět vykreslovat 3 typy grafů*
PK006	*Aplikace bude podporovat přidávání a mazání uživatelských ukazatelů a jejich vykreslení do grafu*
PK007	*Aplikace bude podporovat alespoň dva typy technických ukazatelů*
PK008	*Aplikace bude poskytovat zabezpečené webové služby pro napojení grafů na databázi*
PK009	*Aplikace bude poskytovat zabezpečené webové služby pro připojení mobilních klientů k databázi*

5.2.2 Nefunkční požadavky

Tabulka 5.4 Hlavní nefunkční požadavky kladené na aplikaci AkcieWebKlient

Značení požadavku	*Popis*
PK001	*Aplikace bude mít vygenerovánu dokumentaci a kód aplikace bude okomentován*
PK002	*Uživatelské rozhraní bude v českém jazyce*
PK003	*Aplikace bude vyžadovat .NET Framework 4.0 (a vyšší) a IIS 7.0 (a vyšší)*

5.2.3 Třídní diagram



Obr. 5.3 Hlavní třídy aplikace AkcieWebKlient

5.3 Návrh mobilní aplikace AkcieMobilKlient

Mobilní aplikace by měla využívat rámce zabezpečení webové aplikace pro ověření a přihlášení uživatelů a webové služby poskytnuté webovou aplikací pro přístup k databázi.

5.3.1 Funkční požadavky

Tabulka 5.5 Hlavní funkční požadavky kladené na aplikaci AkcieMobilKlient

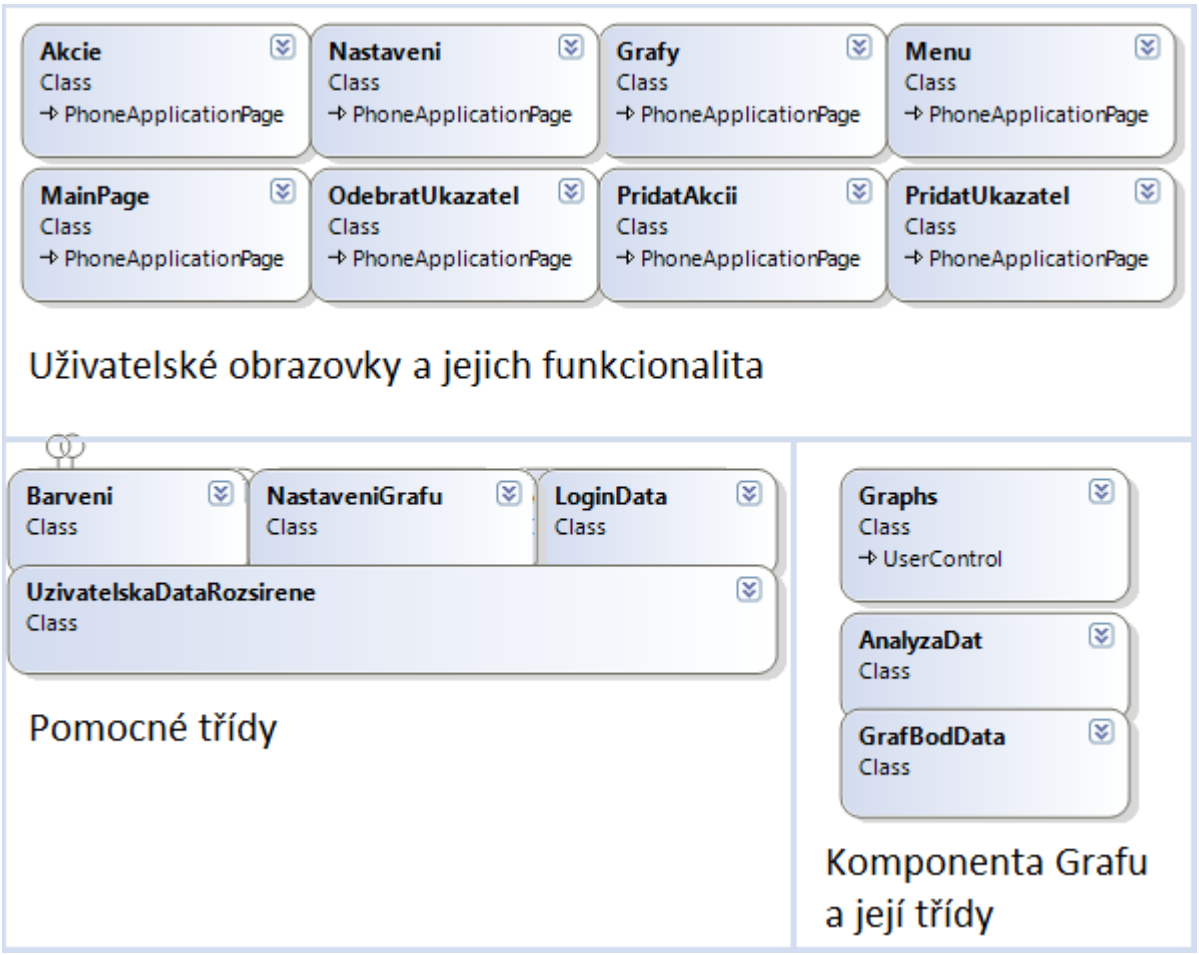
Značení požadavku	*Popis*
PK001	*Aplikace umožní přihlašování uživatelů dle stejného jména a hesla jako webová aplikace*
PK002	*Aplikace bude podporovat jednoduchou správu portfolia akcií*
PK003	*Aplikace umožní uživatelům vést informace o nákupech a prodejích akcií*
PK004	*Aplikace bude umět vykreslovat 3 typy grafů*
PK005	*Aplikace bude podporovat přidávání a mazání uživatelských ukazatelů a jejich vykreslení do grafu*
PK006	*Aplikace bude podporovat alespoň dva typy technických ukazatelů*
PK007	*Aplikace bude přistupovat k datům v databázi pomocí zabezpečených webových služeb poskytovaných webovým klientem*
PK008	*Aplikace bude optimalizována na ovládání dotykem*

5.3.2 Nefunkční požadavky

Tabulka 5.6 Hlavní nefunkční požadavky kladené na aplikaci AkcieMobilKlient

Značení požadavku	*Popis*
PK001	*Aplikace bude mít vygenerovánu dokumentaci a kód aplikace bude okomentován*
PK002	*Uživatelské rozhraní bude v českém jazyce*
PK003	*Aplikace bude pro svůj provoz vyžadovat Windows Phone 7.5 nebo vyšší*

5.3.3 Třídní diagram



Obr. 5.4 Hlavní třídy aplikace AkcieMobilKlient

V návrhu jsme si ukázali schéma navrhnuté databáze, jednotlivé hlavní funkční a nefunkční požadavky všech vyvíjených aplikací a taktéž diagramy tříd těchto aplikací. V následující kapitole si přiblížíme samotnou implementaci.

6 Implementace serverové části

V této kapitole si přiblížíme implementaci aplikace běžící na serveru. Shrneme si zde nejdůležitější využité třídy a poté si blíže ukážeme samotnou aplikaci včetně jejího vzhledu a jednotlivých funkcí.

6.1 Použité třídy

Celá aplikace obsahuje třídy uživatelského rozhraní, dále třídy pro práci s databází a XML a nakonec třídy pro aplikační logiku (převody, ukládání výpisů aplikace apod.) jak již bylo zobrazeno na třídním diagramu v kapitole návrhu. Nyní si tyto třídy přiblížíme.

6.1.1 Třídy pro práci s GUI

Následující třídy se starají o zobrazení grafického uživatelského rozhraní aplikace a jeho funkcionalitu. Funkcionalitou je zde myšleno ošetření událostí (kliknutí na tlačítko, změna obsahu textového pole apod.), dále ověřování hodnot zadaných uživatelem a výpisy důležitých informací na obrazovku (chyby, upozornění atd.):

- HlavniOkno
- Napoveda
- Oprogramu
- PridaniAkcii
- PridaniZdrojeDat

Třída HlavniOkno navíc obsahuje metody, kterými lze přistupovat na GUI definované touto třídou z jiných vláken. Ty jsou zde z důvodu výpisů událostí na obrazovku.

6.1.2 Třídy pro práci s daty

Zde patří dvě velice důležité třídy a to:

- PraceSdatabazi
- UlozisteXML

Třída PraceSdatabazi obsahuje metody, které jsou využity pro komunikaci s databází. Jsou zde metody pro kompletní vytvoření databáze na daném serveru, včetně procedur a naplnění daty výchozími akciemi (jedná se o 100 nejobchodovanějších akcií na burze NASDAQ). Taktéž se zde nacházejí metody pro práci s databázovou tabulkou Akcie a HistorickeHodnoty, tedy metody umožňující vkládání hodnot, mazání hodnot, editaci hodnot a načtení hodnot nejrůznějších položek z těchto databází. Poté jsou zde metody pro zpracování stažených hodnot a jejich uložení do databáze (např. metoda, která z důvodu zvýšení efektivity nejprve načte všechny akcie z databázové tabulky obsahující historické hodnoty s nejnovějším datem dle jejich zkratky a poté jejich datum porovná s aktuálním dnem a vrátí dvojice zkratka a číslo, kde číslo znamená počet řádků od data dané akcie až po aktuální den, tudíž se poté nemusí stahovat všechna historická data, ale pouze tolik, kolik

je potřeba). Tato třída je nesmírně důležitá, neboť metody v ní obsažené jsou zodpovědné za plnění databáze akciovými daty.

Dále zde máme třídu UlozisteXML, která je zodpovědná za práci s XML soubory nastavení. XML jsem zvolil pro ukládání nastavení z několika důvodů. Prvním důvodem je, že si lze libovolně definovat jeho strukturu. Druhou výhodou je, že v případě nutnosti, lze pomocí transformací soubor v tomto formátu převést bez problémů do formátu jiného. Tato třída obsahuje metody pro vytváření, ukládání a mazání dat v XML souborech nastavení. Taktéž obsahuje metody pro jednoduché zašifrování či odšifrování hesla pro přístup na MS SQL server a jeho následné uložení do souboru. Vzhledem k tomu, že aplikace bude běžet na serveru, který by měl sám o sobě býti zabezpečen, je zde toto šifrování pouze z důvodu zvýšení bezpečnosti.

6.1.3 Třídy s aplikační logikou

Do této kategorie patří ostatní třídy nacházející se v programu, které mají na starost další věci spjaté s funkčností aplikace. Jsou to tyto třídy:

- DownloadedData
- Dvojice
- Sloupce
- Server
- Stahovani
- Logovani
- PrevodyOverovani

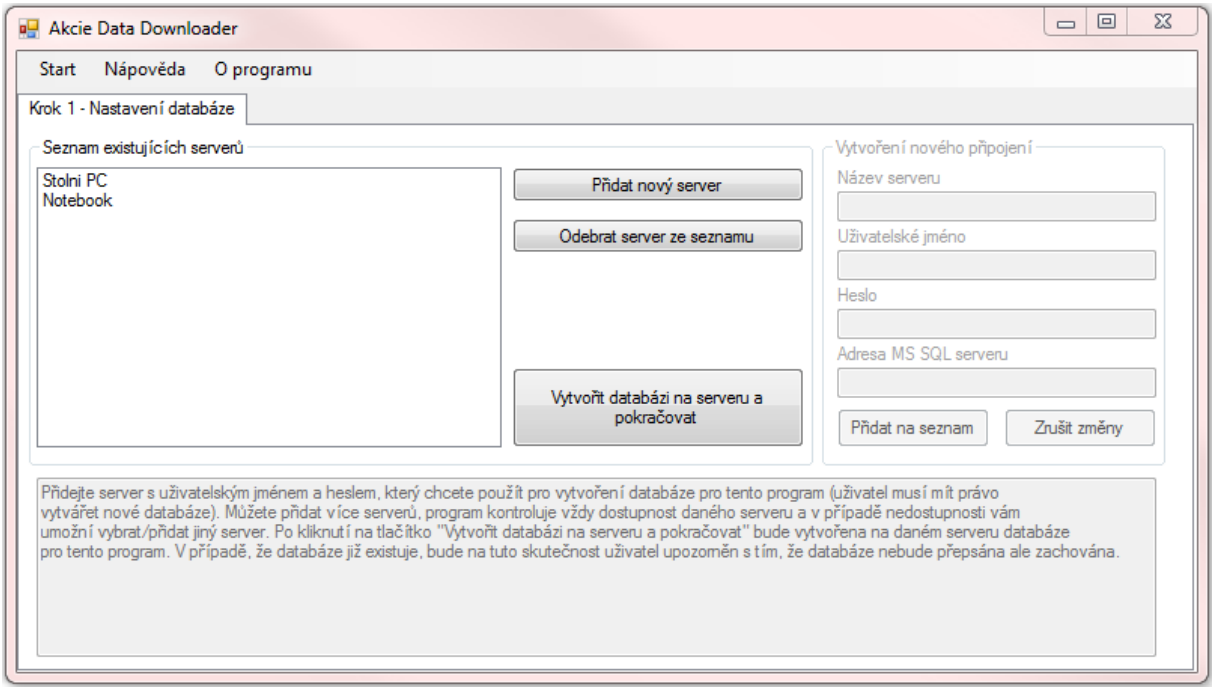
První čtyři třídy slouží k reprezentaci dat. Třída DownloadedData uchovává informace o nastavení historických zdrojů či zdrojů v reálném čase. Třída Dvojice uchovává informace o názvu akcie a její zkratce a umožňuje tyto hodnoty vypsat díky přetížení metody ToString() v uživatelsky přívětivém tvaru. Třída Sloupce uchovává informace o přiřazení čísla sloupce a hodnoty neobchodování k názvu sloupce (např. Open, 1, N/A znamená název sloupce Open, který je ve zdroji na 1 pozici a v případě neobchodování jeho hodnota nabývá hodnoty N/A) a taktéž umožňuje vypsat dané hodnoty v uživatelsky přívětivém tvaru. Nakonec je zde třída Server, která uchovává informace nutné pro přístup na databázový server.

Zbývající tři třídy poté obsahují další funkcionalitu. Třída Stahovani slouží ke stahování dat. Jsou zde obsaženy metody pro stažení dat a jejich předzpracování. Důležité je zde zmínit, že jsou zde metody využívající technologie Task Parallel Library pro stahování dat paralelně. Tím je docíleno větší efektivity aplikace a hlavně zrychlení samotného stahování, kdy je několik dat akcií stahováno zároveň. Další třídou je třída Logovani, která se stará o ukládání informací o běhu programu do souboru. Pro každý den je vytvořen nový soubor a vzhledem k tomu, že stahování dat probíhá paralelně, je do tohoto souboru umožněn přístup z více vláken. Nakonec je zde třída PrevodyOverovani, která má na starost ověřování a převody různých typů dat.

Tímto jsme si zjednodušeně přiblížili jednotlivé důležité třídy. Pro více informací o daných třídách a hlavně metodách, které poskytují, doporučuji podívat se do dokumentace nebo přímo do okomentovaného zdrojového kódu na přiloženém disku CD.

6.2 Aplikace a její funkce

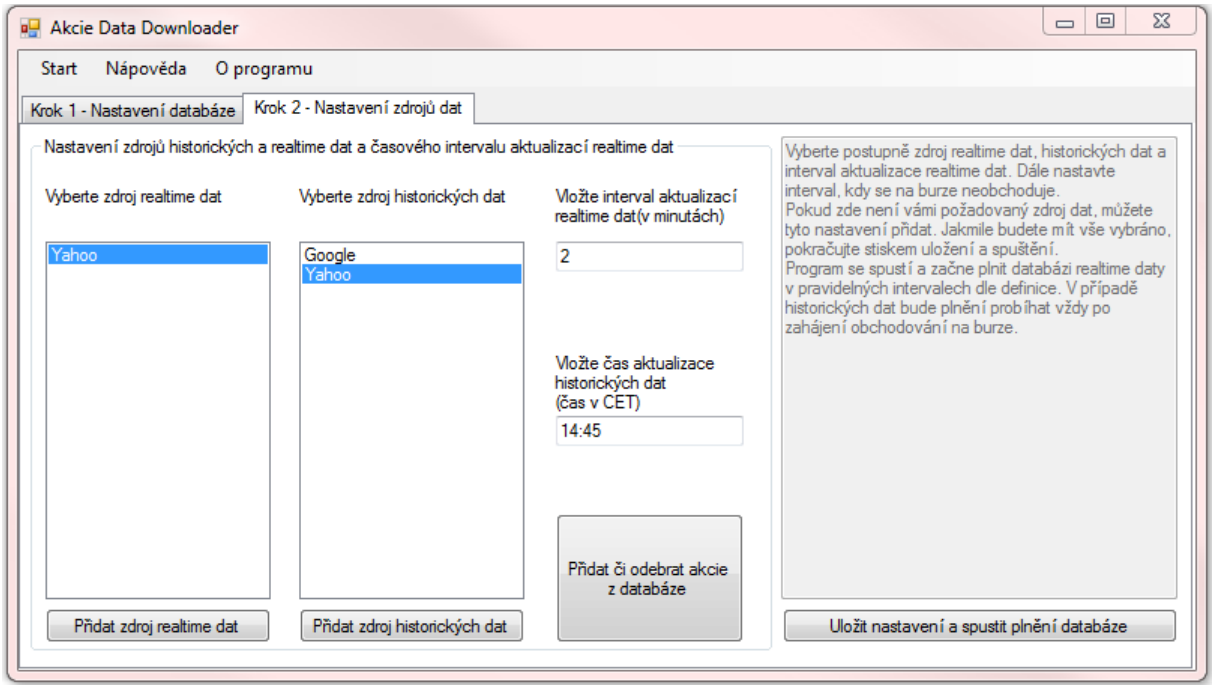
Nyní se dostáváme k samotné aplikaci a její funkcionalitě. Vzhledem k tomu, že aplikace poběží na serveru, bude se o její chod starat administrátor daného serveru. Po spuštění aplikace musí administrátor nejprve zvolit MS SQL server, který chce použít pro vytvoření databáze. Pokud na seznamu není žádný server přítomen, musí ho administrátor nejprve přidat. Jakmile bude s tímto hotov, aplikace ověří, zdali je vše v pořádku, a pokud ano tak přidá server na seznam (Obr. 6.1).



Obr. 6.1 Úvodní obrazovka aplikace s možností výběru a definice MS SQL serveru pro vytvoření databáze

Poté již stačí tento server vybrat ze seznamu a pokračovat stiskem tlačítka Vytvořit databázi na serveru a pokračovat. Po stisknutí tohoto tlačítka dojde k vytvoření databáze včetně všech procedur a vložení 100 nejobchodovanějších akcií na burze NASDAQ do databáze. Pokud již databáze existuje, bude na toto administrátor upozorněn a bude tato existující databáze použita. Pokud vše proběhne bez problémů, přejde se na záložku krok 2 (Obr. 6.2). Zde musí administrátor zvolit jednotlivé zdroje dat (tedy jak zdroj historických dat, tak zdroj dat v reálném čase). Jakmile toto udělá, musí ještě definovat intervaly aktualizace jednotlivých dat. Historická data se aktualizují jednou denně a to vždy v čas zadaný administrátorem. Data v reálném čase jsou poté aktualizována dle zadaného intervalu, který může nabývat hodnot od 1 do 300 minut. Samozřejmostí je, že administrátor může definovat také vlastní zdroje dat. Tedy díky tomu může aplikace stahovat data libovolné burzy, pro kterou ovšem musí být dostupný nějaký veřejný zdroj dat. Tyto zdroje pak lze definovat nezávisle jak pro data v reálném čase (po kliknutí na Přidat zdroj realtime dat) tak pro historická data (po kliknutí na Přidat zdroj historických dat). Pokud ovšem chceme použít zdroj dat pro jinou burzu než výchozí NASDAQ, budeme potřebovat odebrat výchozí akcie z databáze a přidat akcie, které se obchodují na dané burze. Toto je zde samozřejmě umožněno, stačí kliknout na tlačítko Přidat či odebrat akcie z databáze. Jakmile jsou všechny věci úspěšně nastaveny, klikneme na tlačítko Uložit nastavení a spustit plnění

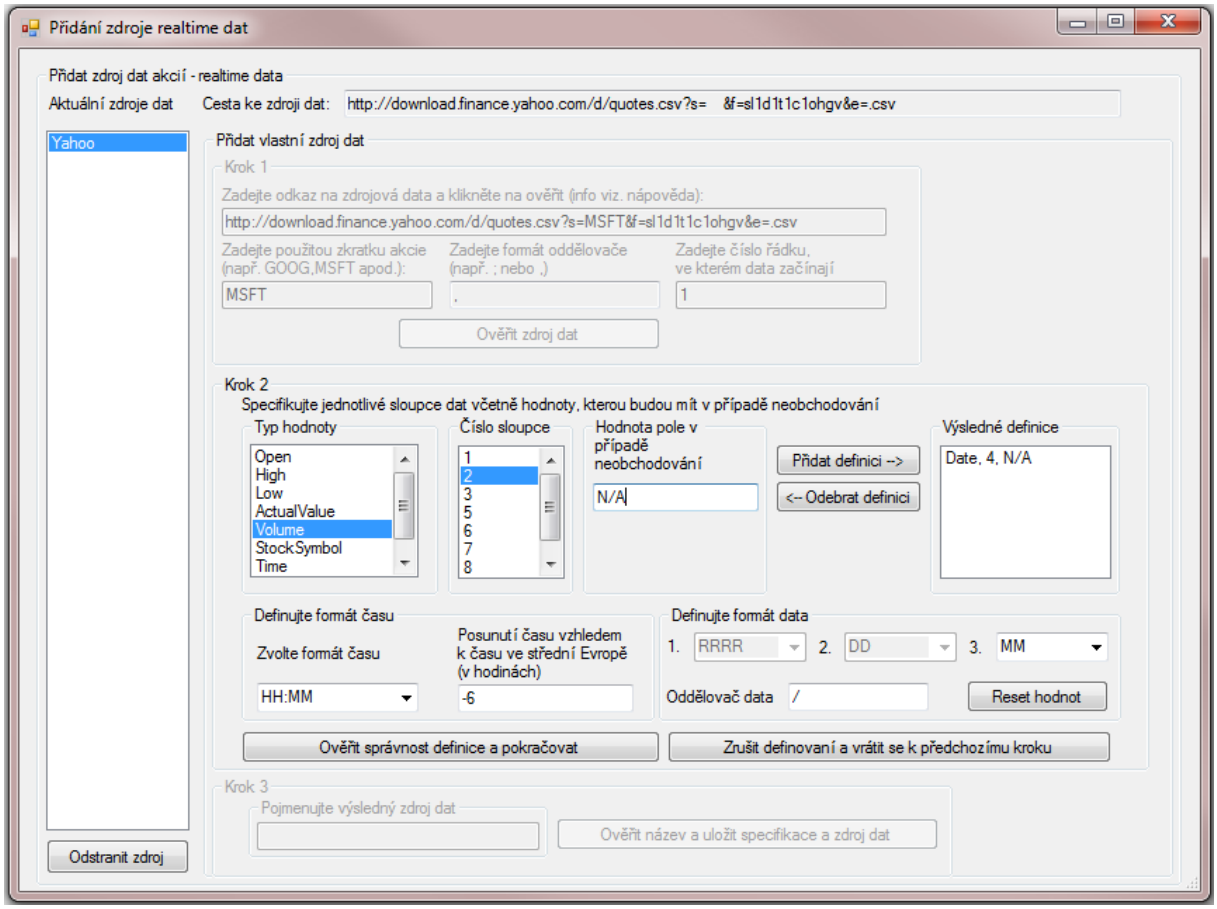
databáze a začne probíhat plnění. K tomu se dostaneme později. Nyní si přiblížíme definování zdroje dat a práci s akciemi v databázi.



Obr. 6.2 Záložka Krok 2, na které administrátor vybere zdroje dat a zadá intervaly aktualizace dat

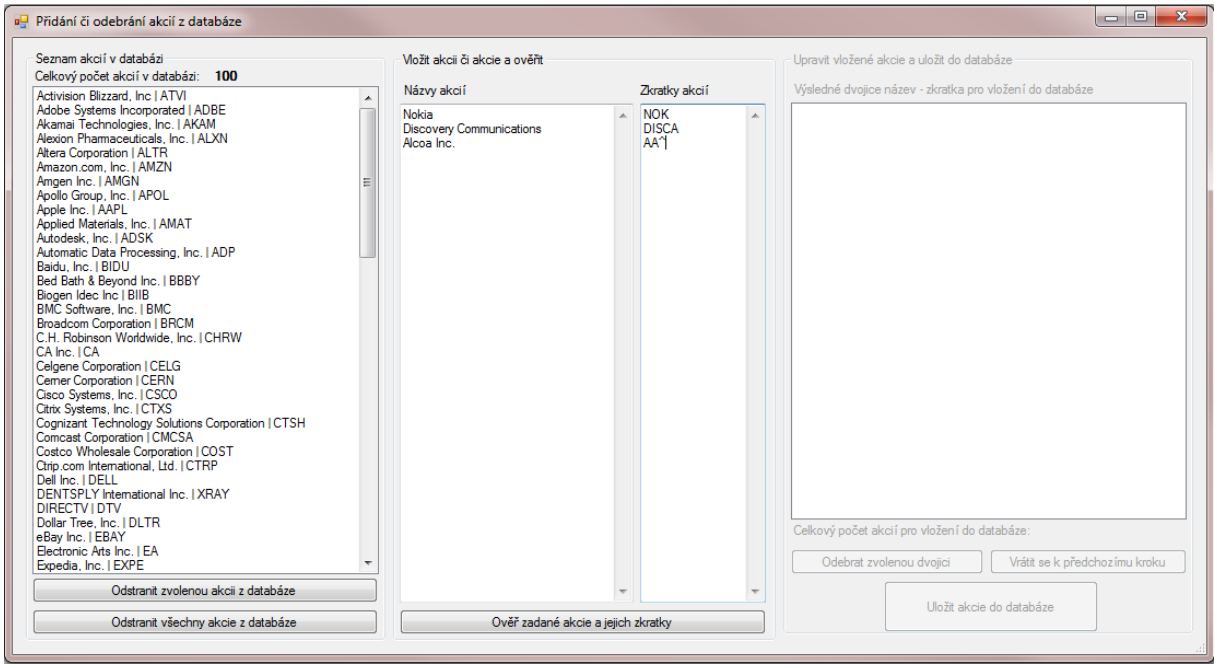
Definování zdroje dat je pro tuto aplikaci klíčové, neboť kromě výhody v podobě sledování libovolné burzy, přináší také možnost zvolit jiný zdroj dat v případě nefunkčnosti aktuálního zdroje dat. To sebou přináší samozřejmě řadu výhod. Jakmile zdroj dat přestane fungovat, administrátor zastaví plnění databáze, změní zdroj dat a znovu spustí plnění. Stávající data tedy není potřeba zahodit. Nová data se začnou přidávat do aktuální databáze, aniž by klienti poznali nějaký rozdíl. Na obrázku (Obr. 6.3) je zobrazen formulář pro definici zdroje dat. Administrátor musí nejprve zadat webovou adresu daného zdroje. Počítá se s tím, že se bude jednat o nějaký typ textového souboru, kdy jednotlivé hodnoty budou odděleny nějakým oddělovačem. Administrátor vyspecifikuje, ke které akciové zkratce se zdroj vztahuje (ta bude poté nahrazována jinými zkratkami akcií dané burzy), dále vyspecifikuje oddělovač dat a nakonec přidá číslo řádku, na kterém data začínají (je to z důvodu, že některé zdroje mají na prvním řádku názvy daných hodnot). Jakmile toto vše udělá, dá ověřit zdroj dat a v případě úspěchu může pokračovat v další specifikaci. Nyní musí spojit název hodnoty spolu s číslem sloupce, ve kterém se daná hodnota v datovém souboru nachází, a s hodnotou při neobchodování. Je to z důvodu, že různé zdroje se pořadím sloupců liší. Navíc některé zdroje mají sloupců více, což právě toto řeší. Musí být vyčerpány všechny hodnoty v seznamu Typ hodnoty, aby aplikace úspěšně ověřila správnost. Hodnota při neobchodování znamená hodnotu, kterou daná pole mohou nabývat při neobchodování. Pokud se jedná o stejnou hodnotu jako v případě obchodování (například číslo), administrátor musí ponechat pole prázdné. Tato hodnota je zde ale hlavně kvůli odlišení dat akcie v případě, že se momentálně na burze neobchoduje od případů, kdy daná akcie neexistuje (některé zdroje dat, jako například Yahoo Finance, vracejí v případě neobchodování u některých polí místo čísla hodnotu N/A, tato hodnota je poté u všech polí v případě neexistence akcie). Jakmile administrátor vydefinuje všechny sloupce, musí ještě vydefinovat datum a čas. Čas

může být buď ve 24 hodinovém formátu, nebo ve formátu 12 hodinovém s koncovkou am (dopoledne) či pm (odpoledne). Dále musí zadat posunutí času vzhledem k času ve střední Evropě (od -12 hodin do 12 hodin). Co se data týče, je potřeba definovat kompletní formát data a v jakém pořadí se objevuje rok, den, týden v daném datu a jakým oddělovačem jsou odděleny. Jakmile je vše nadefinováno, může administrátor přejít k ověření správnosti definicí kliknutím na tlačítko Ověřit správnost definice a pokračovat. Pokud je dosaženo úspěchu, pojmenuje ještě zdroj dat a nechá ho uložit kliknutím na Ověřit název a uložit specifikace a zdroj dat. Tímto byl úspěšně nadefinován nový zdroj dat.



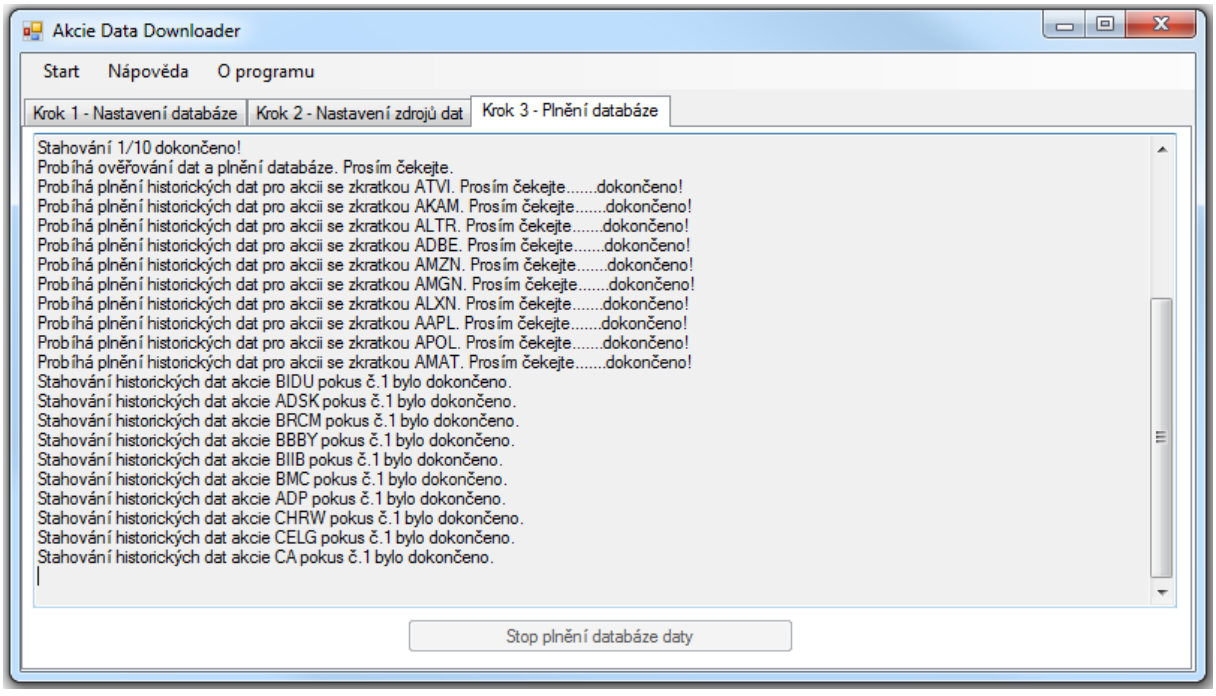
Obr. 6.3 Formulář pro definování zdroje dat v reálném čase

Jak již bylo řečeno výše, aplikace umožňuje taktéž práci s akciemi v databázi. Na obrázku 6.4 je zobrazen formulář, který má toto za úkol. V seznamu úplně nalevo se nachází akcie (dvojice název firmy a zkratka), které jsou již přítomné v databázi. Zde může administrátor smazat vybranou akcii či smazat všechny akcie z databáze (například pokud chce využít akcie jiné burzy). Uprostřed poté může definovat akcie nové. Každý název a každá zkratka se musí nacházet na novém řádku. Počet řádků s názvy musí odpovídat počtu řádků se zkratkami. Prázdné řádky jsou samozřejmě při zpracování vynechány. Jakmile administrátor definuje všechny akcie, které chce do databáze přidat, stiskne tlačítko Ověř zadané akcie a jejich zkratky. Po stisknutí dojde k ověření dat. Veškeré akcie, které již v databázi jsou, budou z definovaných dat odstraněny a vše zbývající, co projde kontrolou, bude zobrazeno (opět jako dvojice název firmy a zkratka) v seznamu napravo. Zde může administrátor udělat poslední úpravy, nehodící se akcie odstranit, a jakmile bude vše hotovo, stiskem tlačítka Uložit akcie do databáze akcie uloží. Tímto lze tedy měnit obsah akcií v databázi dle potřeb.



Obr. 6.4 Formulář pro editaci akcií v databázi

Nyní se můžeme vrátit k samotnému plnění databáze. Jakmile administrátor vše definuje a spustí plnění, budou se mu na obrazovku vypisovat údaje o průběhu (Obr. 6.5).



Obr. 6.5 Formulář zobrazující průběh plnění databáze

Tento formulář může minimalizovat, aby ho nerušil a v případě nutnosti opět vyvolat kliknutím na malou ikonu vedle hodin v systému Windows. Některé údaje, hlavně tedy chyby, jsou ukládány do textových souborů. Název souboru je vždy aktuální datum a soubory jsou ukládány ve složce, kde se nachází samotná aplikace. Samotné stahování poté probíhá tak, že se nejprve spustí stahování dat akcií v reálném čase, aby byly zjištěny akcie, které neexistují a odstranily se. Poté se

spustí stahování historických dat akcií, a jakmile je vše úspěšně staženo, jsou vytvořena dvě vlákna – jedno pro stahování dat v reálném čase a jedno pro stahování dat historických hodnot. Tyto vlákna tedy pracují nezávisle na sobě a aktualizují databázi, dle intervalů zadaných na obrázku (Obr. 6.2). Administrátor může kdykoliv, když zrovna nedochází k plnění databáze, přerušit práci aplikace.

Tímto jsem zde shrnul veškeré funkce této aplikace. Samozřejmě její existence je pro ostatní dvě aplikace klíčová. Bez ní by totiž neexistovala nezávislost na zdroji dat a v případě zrušení jednoho zdroje, by klientské aplikace musely být předělány, aby opět fungovaly. Zde stačí, aby byla zajištěna funkčnost serveru, na kterém aplikace běží, a vše bude fungovat i po změně zdroje dat tak jak má. K této aplikace existuje také vývojářská dokumentace, která je součástí přiloženého disku CD, na kterém lze nalézt taktéž zdrojový kód aplikace.

7 Implementace klientské části - Webový klient

V této kapitole si přiblížíme implementaci první ze dvou klientských aplikací a to webového klienta. Shrneme si zde nejdůležitější využité třídy a poté si blíže ukážeme samotnou aplikaci včetně jejího vzhledu a jednotlivých funkcí.

7.1 Použité třídy a komponenty

Každý webový formulář ASP.NET je svázán se třídou, jež se stará o jeho funkcionalitu (tzv. CodeBehind). V těchto třídách se nachází vše, co je potřeba pro danou stránku, tedy práce s daty, ověřování hodnot a práce s uživatelským rozhraním. Aplikace dále obsahuje třídy spjaté s komponentou Captcha a třídy spjaté s webovými službami. To vše bylo představeno na digramu tříd v kapitole 5. Nicméně aplikace využívá další důležité komponenty a to grafy, postavené na technologii Silverlight a custom Membership Providera, který se stará o správu uživatelů. Nyní si tedy tyto části blíže rozebereme.

7.1.1 Třídy starající se o webové formuláře poskytující zabezpečení

Tyto třídy se starají o zabezpečení webového klienta. Všechny využívají custom Membership Providera (o něm později) definovaného aplikací. Patří sem:

- Login
- Registrace
- ZiskaniHesla
- ZmenaHesla

Webový formulář Login již obsahuje komponentu, která se automaticky postará o přihlášení uživatele za použití aplikací definovaného Membership Providera. Samozřejmostí je přítomnost ověření zadaných hodnot v jednotlivých polích.

Třída Registrace, která se stará o funkcionalitu webového formuláře stejného názvu, má za úkol registraci uživatele. Webový formulář taktéž obsahuje ověření zadaných hodnot v jednotlivých polích. Jakmile je vše ověřeno, Membership Provider vytvoří uživatele. Pokud vše proběhne správně, bude daný uživatel přihlášen, pokud se objeví chyba, bude uživatel informován.

Třída ZiskaniHesla se stará o obnovení zapomenutého hesla. Obnovení probíhá na základě znalosti odpovědi na uživatelem zadanou bezpečnostní otázku, kterou vložil při registraci. V případě úspěchu vygeneruje Membership Provider nové heslo, které by si měl uživatel později změnit.

Třída ZmenaHesla se stará o změnu hesla přihlášeného uživatele. Uživatel zadá do webového formuláře stejného názvu své staré heslo, poté dvakrát nové heslo a potvrdí. Membership Provider se následně postará o změnu hesla.

7.1.2 Třídy starající se o ostatní webové formuláře

Tyto třídy se starají o funkcionalitu ostatních webových formulářů. Patří sem:

- About

- Default
- Napoveda
- SiteMaster
- Grafy
- Portfolio

První tři třídy nemají žádnou funkcionalitu navíc, neboť jejich webové formuláře pouze zobrazují text (informace o aplikaci, úvodní stránka a nápověda). Třída SiteMaster nemá taktéž žádnou funkcionalitu navíc. Její webový formulář Site.Master obsahuje komponentu, která se stará o zobrazení přihlášeného uživatele. Navíc se jedná o speciální formulář, kterého ostatní formuláře využívají. Lze si jej představit jako jakousi vzorovou stránku, kterou poté sdílí ostatní stránky. Tudíž není potřeba definovat design a funkcionalitu, která má být sdílena, pro každou stránku zvlášť. Ovládací prvky, které sdíleny být nemají a jsou unikátní pro každý webový formulář, jsou vkládány do tzv. ContentPlaceHolder.

Třída Grafy je již ale velmi důležitá, neboť se stará o správnou funkci grafů. Umožňuje načtení seznamu akcií daného uživatele z databáze a jejich zobrazení. Při načítání stránky následně nastaví grafové komponentě výchozí parametry (první akcie v seznamu akcií daného uživatele), které obsahují zkratku akcie a uživatelské jméno, aby byla grafová komponenta schopna stahovat data z databáze. Při změně výběru akcie změní také tyto parametry. Tím je zajištěna interakce mezi webovým formulářem a grafovou komponentou v Silverlightu.

Poslední třída Portfolio, která se stará o funkcionalitu stejnojmenného webového formuláře, obsahuje všechny metody potřebné pro zobrazení uživatelských akcií, práci s uživatelskými akciemi (mazání či zobrazení upřesňujících informací vybrané akcie) a přidání akcií mezi uživatelské akcie. Dále jsou zde metody pro přidávání nákupů či prodejů akcií do databáze, výpočet zisku a přepočet zisku dle uživatelem zadaného kurzu. Veškeré formuláře jsou před zpracováním ověřeny. Taktéž je zde zamezeno použití technik, které by mohly umožnit napadení databáze, použitím parametrizovaných SQL dotazů. Pro seznam všech metod a funkcí doporučuji nahlédnout do vývojářské dokumentace či okomentovaného zdrojového kódu na přiloženém disku CD.

7.1.3 Captcha komponenta a její třídy

Z důvodu zajištění větší bezpečnosti, tedy hlavně znemožnění snahy opakovat pokusy o přihlášení, registraci či změnu hesla jsem zahrnul do webové aplikace taktéž Captcha komponentu. K napsání této komponenty jsem využil návodu, který je k dispozici zde [17]. Návod byl napsán ve Visual Basicu, takže bylo nutno celou komponentu předělat do jazyka C#. Tato komponenta je celkem jednoduchá a určitě by se na ní daly některé věci vylepšit, nicméně pro stávající použití postačí. Třídy, které komponenta využívá, jsou následující:

- Captcha
- CaptchaKonfigurace
- CaptchaObrazek
- CaptchaValidator
- KomponentyKonfigurace

Třída Captcha obsahuje metody nutné pro fungování samotné komponenty. CaptchaKonfigurace obsahuje konfigurační vlastnosti komponenty. CaptchaObrazek se stará o práci s obrázkem, ve kterém bude text pro opsání. CaptchaValidator se stará o ověření, zdali byl opsaný text roven vygenerovanému textu a nakonec KomponentyKonfigurace je konfigurační sekcí Captcha komponenty. Captcha samotná je poté definována a nastavena pro použití v aplikace v souboru web.config.

7.1.4 Webové služby a jejich třídy

Vzhledem k tomu, že aplikace postavené na technologii Silverlight neumí přistupovat k databázi přímo, ale pouze za využití webových služeb, bylo potřeba vytvořit zabezpečené webové služby poskytující metody pro přístup k databázi. Celkově se jedná o tři webové služby. Jedna je zde pro přístup grafů k databázi a zbývající dvě jsou určeny pro mobilního klienta. Kromě těchto webových služeb jsou zde další třídy reprezentující data. Webové služby jsou tedy následující:

- WindowsPhoneAuthentication
- WindowsPhoneService
- SilverlightGraphs

Služba WindowsPhoneAuthentication je jedinou ze služeb, která je viditelná pro nepřihlášené uživatele a jejím úkolem je ověřit přihlašovací údaje uživatele mobilního klienta. Využívá přitom služeb zabezpečení webové aplikace. Informace o přihlášení jsou na mobilním klientovi uchovávány v podobě cookies. Aby toto fungovalo, musí být na mobilním klientovi povolen Cookie Container.

Služba WindowsPhoneService je službou, ke které mohou přistupovat pouze přihlášení uživatelé mobilního klienta. Funguje to tak, že při každém volání metody této služby z mobilního klienta dochází k odeslání autentizační cookie, díky které pak může služba povolit přístup. Třída této služby implementuje rozhraní IWindowsPhoneService a obsahuje metody, které umožňují mobilní aplikaci komunikaci s databází.

Nakonec zde máme službu SilverlightGraphs, která se stará o dostupnost dat pro grafy. Třída této služby implementuje rozhraní ISilverlightGraphs a poskytuje metody, které umožňují grafové komponentě přístup k potřebným datům v databázi. Zde se jedná především o přístup k historickým datům zvolené akcie, která mají být vykreslena do grafu a taktéž přístup k datům uživatelských ukazatelů.

Dále jak jsem již zmínil, jsou zde třídy reprezentující data. Tyto třídy jsou serializovatelné (jejich instance lze převést na Stream a např. odeslat pomocí HTTP protokolu) a pracují s nimi webové služby. Mezi tyto třídy patří:

- UzivatelskaData
- PortfolioData
- HistorickaData
- GrafDataPolozka

Třída UzivatelskaData reprezentuje uživatelská data vztahující se k akcii (tedy nákup či prodej akcie a všechny hodnoty s tím spojené). Třída PortfolioData reprezentuje upřesňující informace

o jedné akcii. Třída HistorickaData reprezentuje jednu položku historických dat. A nakonec třída GrafDataPolozka reprezentuje data jednoho uživatelského ukazatele.

7.1.5 Grafová komponenta v Silverlightu

Pro vykreslování grafů ve webové aplikaci jsem využil grafy firmy amCharts [18]. Tyto grafy jsem využil z důvodu jejich volné dostupnosti, kdy jediným omezením je malý odkaz na stránky této firmy v levém horním rohu každého grafu. Pro potřeby aplikace jsem je ale musel upravit a hlavně kompletně napsat třídy umožňující vykreslování uživatelských ukazatelů. Vzhledem k tomu, že se jedná o technologii Silverlight, tak je vzhled definován v souborech XAML. Každý XAML soubor má poté odpovídající soubor s kódem, který se stará o jeho funkcionalitu. Komponenta obsahuje tyto XAML soubory:

- App
- GrafSwitch
- NoDataPage
- CarovyGraf
- OhlcGraf
- SvicnovyGraf

App je hlavní XAML soubor aplikace, nezobrazuje žádné informace sám o sobě, ale jeho soubor s kódem je klíčový, neboť se stará o inicializaci celé aplikace. Do metody, která se vykonává při spuštění aplikace, je přidán kód, který má za úkol stáhnout data nutná pro grafy z databáze a zpracovat je. Stahování probíhá přes webovou službu, ke které jsou grafy připojeny a která je součástí webového klienta. Dále zde máme GrafSwitch, jehož uživatelské rozhraní je prázdné a jehož funkcionalitou je přepínání jednotlivých grafů (jeho obsah se mění v závislosti na zvoleném grafu). NoDataPage se zobrazí v případě, že uživatel nemá žádné akcie ve svém portfoliu nebo se nepovedlo stažení dat. A nyní se dostáváme k samotným XAML souborům grafů (tedy CarovyGraf, OhlcGraf a SvicnovyGraf). Jejich funkcionalita je rozšířena o vkládání a mazání uživatelských ukazatelů. Obsahují tedy metody pro komunikaci s databází, dále metody nutné pro vykreslení různých uživatelských ukazatelů a další pomocné metody. Vše je opět dopodrobna okomentováno přímo v kódu.

Nakonec se zde nacházejí ještě dvě důležité třídy. První třídou je třída Formatovani, která má za úkol konverzi dat. Druhou třídou je poté SampleDataLayer, což je třída reprezentující data pro grafy. Tato třída je nicméně upravena pro zpracování stažených dat. Tyto data zpracuje a poté poskytuje kromě standartních historických dat, nutných pro vykreslení grafů, taktéž data pro vykreslení technických ukazatelů SMA a EMA.

7.1.6 Custom Membership Provider

Při snaze využít zabezpečení pomocí standardního ASP.NET Membership Providera jsem narazil na problém nemožnosti zadat při registraci uživatele více uživatelských informací a taktéž mi nevyhovovaly tabulky, které byly pro uživatele vytvořeny v databázi. Rozhodl jsem se tedy pro úpravu Membership Providera. Využil jsem proto návodu na MSDN [19, 20], který kompletně popisuje tvorbu custom Membership Providera a upravil jsem tento kód tak, aby vyhovoval mé databázi. Byly

tedy přidány položky do databázové tabulky Uzivatele a vytvořeny procedury pro práci s tabulkou Uzivatele. Samotný provider poté obsahuje dvě třídy.

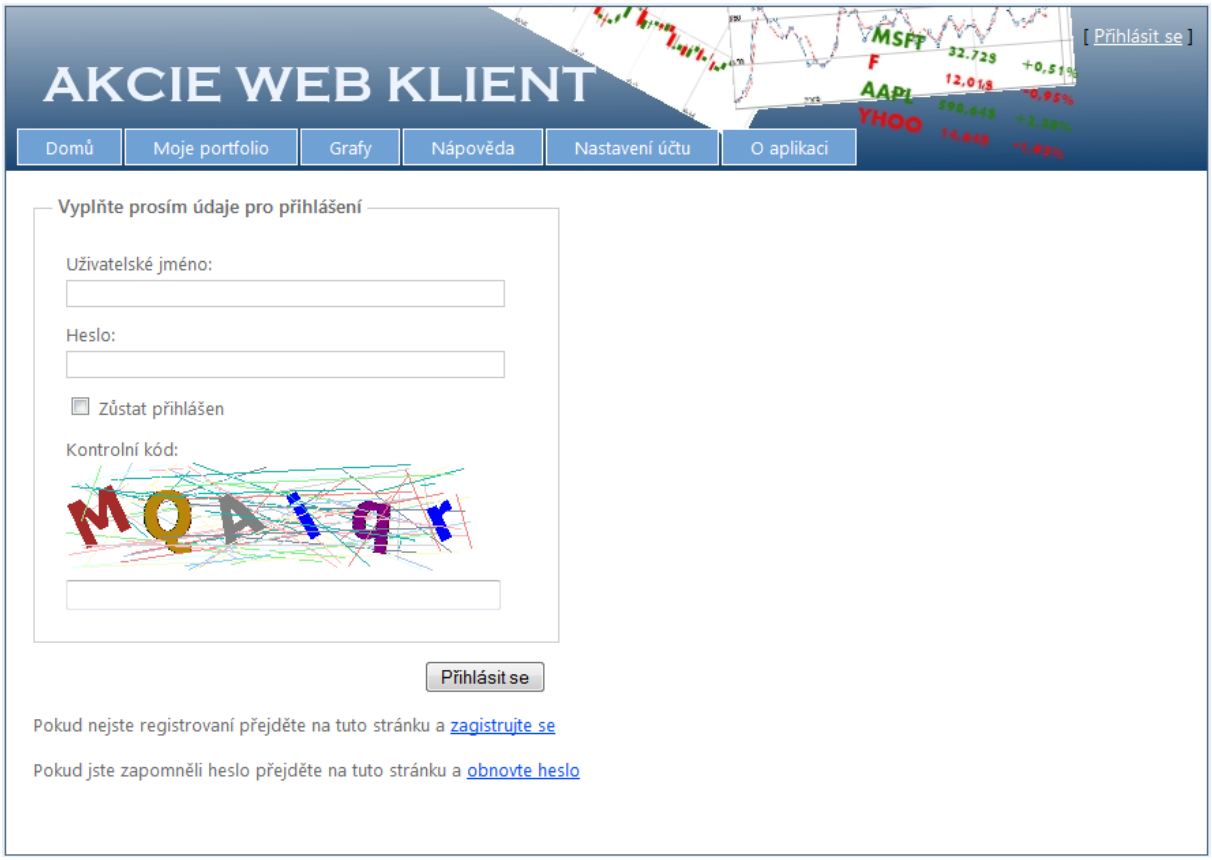
Třída Uzivatel reprezentuje uživatele a obsahuje vlastnosti, které původní ASP.NET Membership Provider neposkytoval.

Třída ZabezpeceniProvider je samotným providerem, který implementuje metody pro vytvoření uživatele, změnu hesla uživatele, smazání uživatele, editaci uživatele, obnovení hesla uživatele atd.

Ne všechny tyto metody jsou v mých aplikacích využity, ale díky tomu je zde prostor pro vylepšování a rozšiřování aplikace.

7.2 Webová aplikace a její funkce

Nyní se již dostáváme k samotné webové aplikaci a její funkcionalitě. Po zadání webové adresy do prohlížeče se uživatel dostane na titulní stranu, na které je zobrazen formulář pro přihlášení (Obr. 7.1). Pokud uživatel ještě není registrován, musí se nejprve zaregistrovat kliknutím na příslušné tlačítko pod přihlašovacím formulářem. V případě, že uživatel zapomněl své heslo, může ho obnovit opět kliknutím na příslušné tlačítko pod přihlašovacím formulářem. Můžete si taktéž všimnout přítomnosti Captcha komponenty. Text v ní obsažen je nutno vždy opsat, jinak se uživatel nebude moci přihlásit.



Obr. 7.1 Titulní stránka webového klienta

V případě, že uživatel kliknul na tlačítko pro registraci, dostal se na stránku s formulářem pro registraci (Obr. 7.2 vlevo). Zde musí vyplnit všechny údaje a poté opět opsat kód z Captchy, aby byla registrace úspěšná. Všimněte si také zobrazeného kalendáře při výběru data narození. Toto vykreslení je zde umožněno díky použití AJAX control toolkitu. AJAX control toolkit obsahuje více jak 30 ovládacích prvků a jedná se o volně dostupný projekt pod open-source licencí, takže jej lze využívat bez omezení.

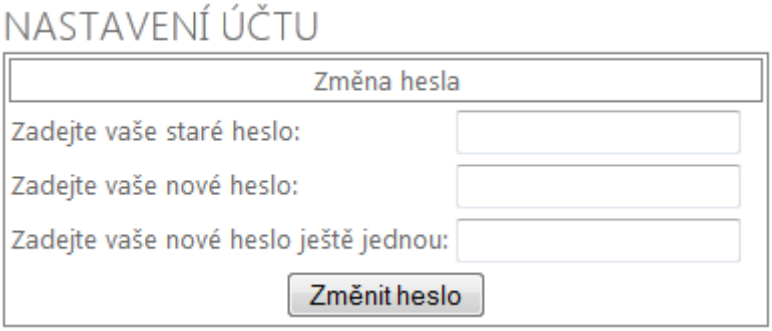
Obr. 7.2 Vlevo formulář pro registraci nového uživatele, vpravo formulář pro obnovení hesla

V případě, že uživatel kliknul na tlačítko pro obnovení hesla, zobrazila se mu stránka s formulářem pro obnovení hesla (Obr. 7.2 vpravo). Zde musí zadat své uživatelské jméno, emailovou adresu a nakonec opsat kontrolní kód. Jakmile vše zadá, nechá si zjistit kontrolní otázku, kterou použil při registraci. Na tuto otázku musí odpovědět stejnou bezpečnostní odpovědí, kterou použil při registraci a poté klikne na příslušné tlačítko pro vygenerování nového hesla. Pokud bude úspěšný, zobrazí se mu okno s jeho novým heslem, které by si měl po přihlášení změnit.

Jakmile je uživatel zaregistrován nebo si obnovil zapomenuté heslo, může pokračovat přihlášením. Po úspěšném přihlášení se dostane na úvodní stránku, která je pouze informativní. Pokud by chtěl uživatel více informací o aplikaci, může kliknout na tlačítko Nápověda. Stejně tak pokud by chtěl zjistit informace o verzi aplikace a tvůrci, může pokračovat stisknutím tlačítka O aplikaci. Tyto stránky mají pouze informativní charakter.

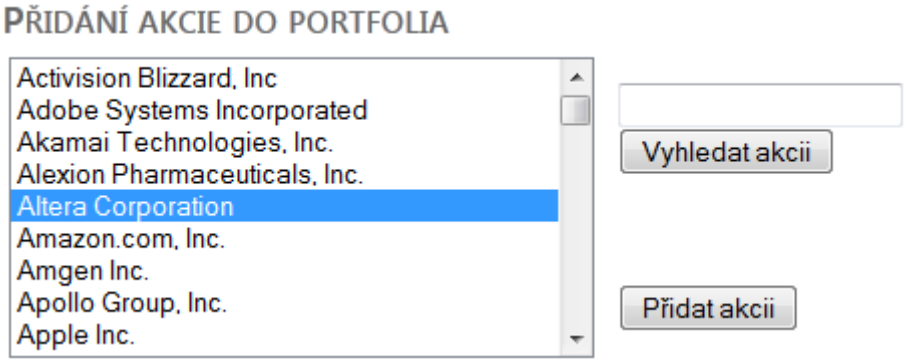
Pokud uživatel obnovil své heslo před přihlášením, je pravděpodobné, že ho bude chtít změnit. Kliknutím na tlačítko Nastavení účtu se mu zobrazí formulář na obrázku (Obr. 7.3). Zde zadá své staré

heslo a poté dvakrát jeho nové heslo (z důvodu ověření). Jakmile je s tímto hotov, stiskne tlačítko Změnit heslo a heslo bude změněno. Všimněte si, že zde chybí komponenta Captcha. Je to z důvodu, že pouze přihlášený uživatel může měnit heslo, takže nutnost použití této komponenty odpadá.



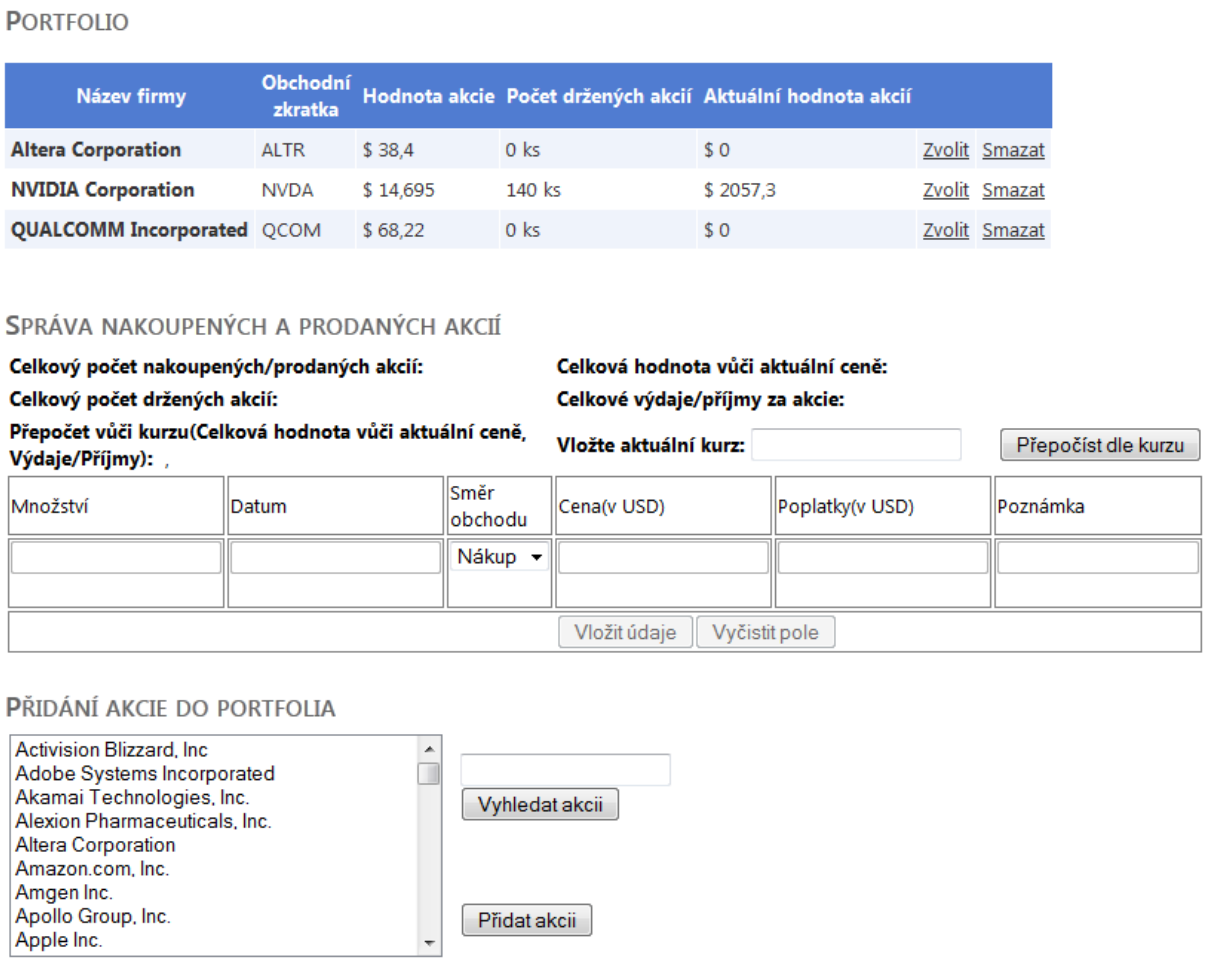
Obr. 7.3 Formulář pro změnu hesla

Nyní se dostáváme k samotné hlavní funkcionalitě aplikace. Ještě předtím, než uživatel navštíví grafy, musí přidat akcie do svého portfolia. Kliknutím na tlačítko Portfolio se uživatel dostane na stránku správy portfolia. Zde nejprve musí ve formuláři Přidání akcie do portfolia přidat akcie (Obr. 7.4).



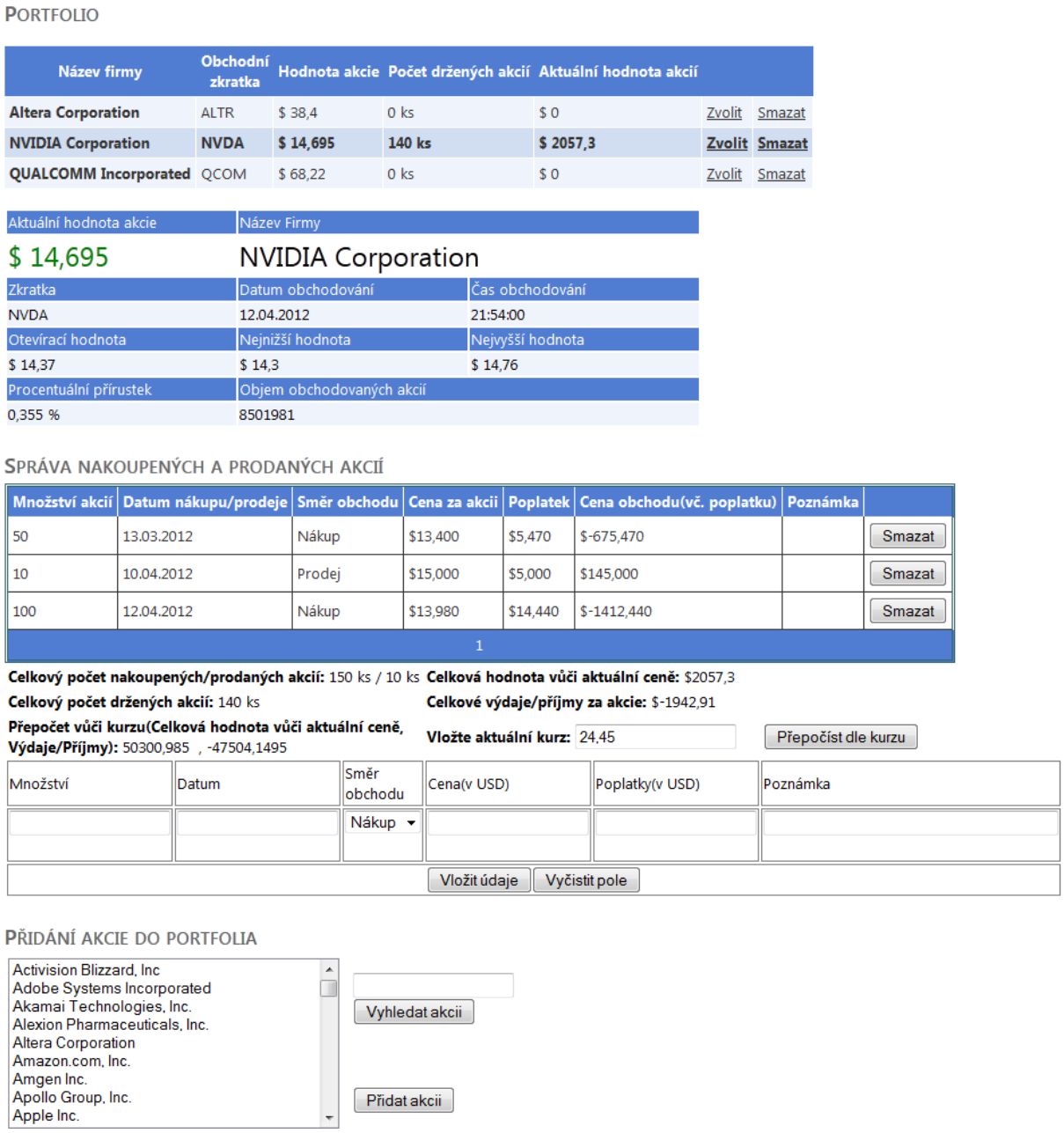
Obr. 7.4 Formulář pro přidání akcie do portfolia

Jakmile uživatel ukončí přidávání akcií, bude zobrazení stránky vypadat nějak takto (Obr. 7.5).



Obr. 7.5 Obsah stránky pro správu portfolia po přidání akcií

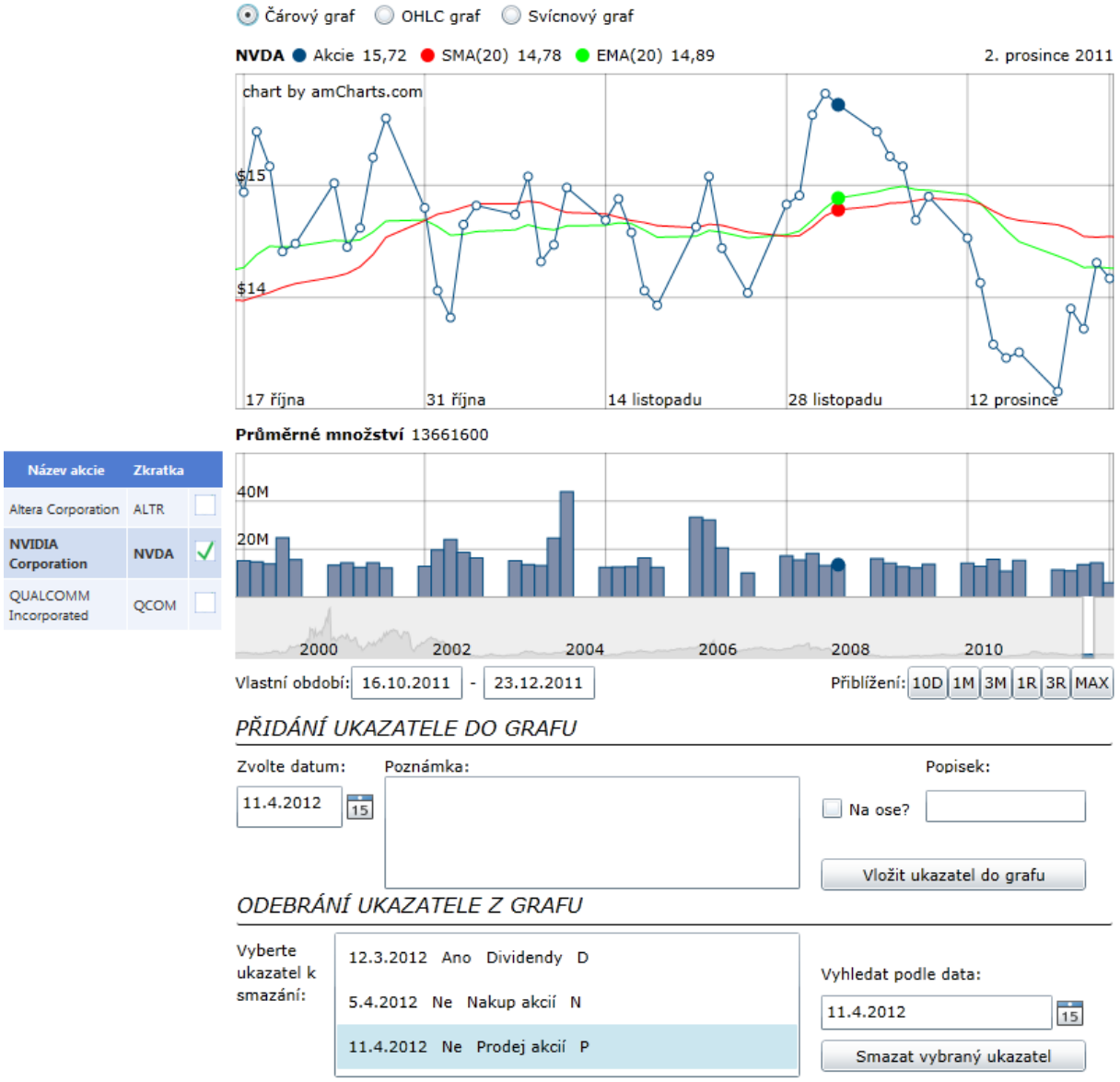
Uživatel může samozřejmě smazat akcie z portfolia či si zobrazit jejich podrobnosti kliknutím na tlačítko Zvolit. Toto tlačítko také aktivuje formulář Správa nakoupených a prodaných akcií. Zde poté uživatel může přidávat informace o nákupu a prodeji akcií zvolené společnosti. Tedy kolik akcií nakoupil či prodal, datum uskutečnění obchodu, cenu jakou stála jedna akcie, poplatky, které zaplatil za uskutečnění obchodu a taktéž může přidat poznámku k danému obchodu. Ve formuláři se mu následně zobrazí kolik akcií má, kolik jich nakoupil a kolik jich prodal. Dále jaká je nyní celková hodnota jeho držených akcií a kolik ho dané obchody s akciemi společnosti stály. Navíc zde může zadat kurz libovolné měny a nechat si dané zisky přepočíst. Jak poté vypadá stránka se zvolenou akcií a seznamem obchodů můžete vidět na obrázku (Obr. 7.6).



Obr. 7.6 Obsah stránky pro správu portfolia se zvolenou společností

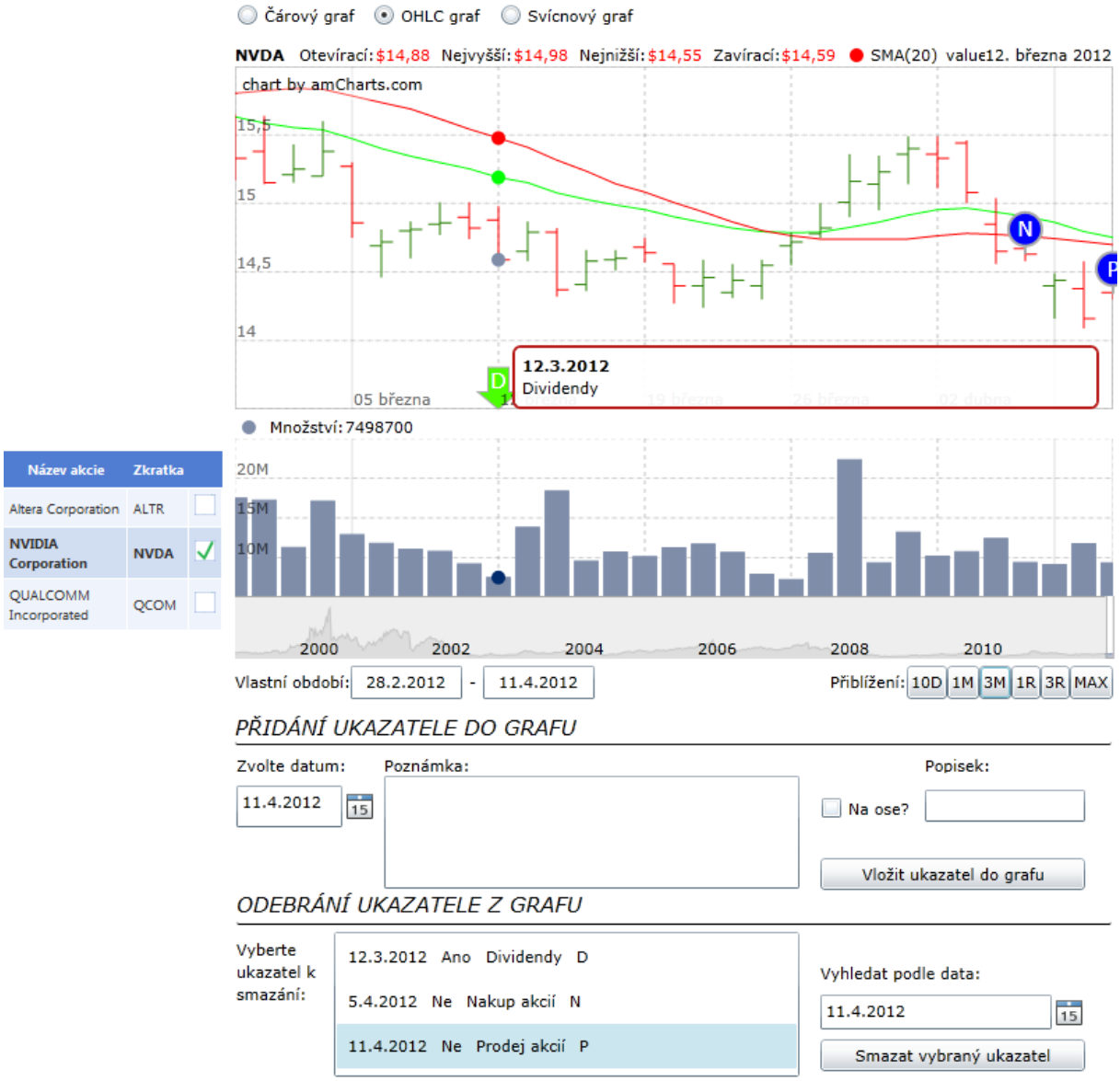
Uživatel si tedy může vést přehledné údaje o svých obchodech nezávisle pro každou akcii. Jakmile již uživatel akcie v portfoliu má, může přejít taktéž na zobrazení grafů.

Jsou zde dostupné tři typy grafů pro zobrazení průběhu vývoje cen akcií. Na obrázku (Obr. 7.7) je zobrazen obsah stránky s grafy.



Obr. 7.7 Obsah stránky s grafy

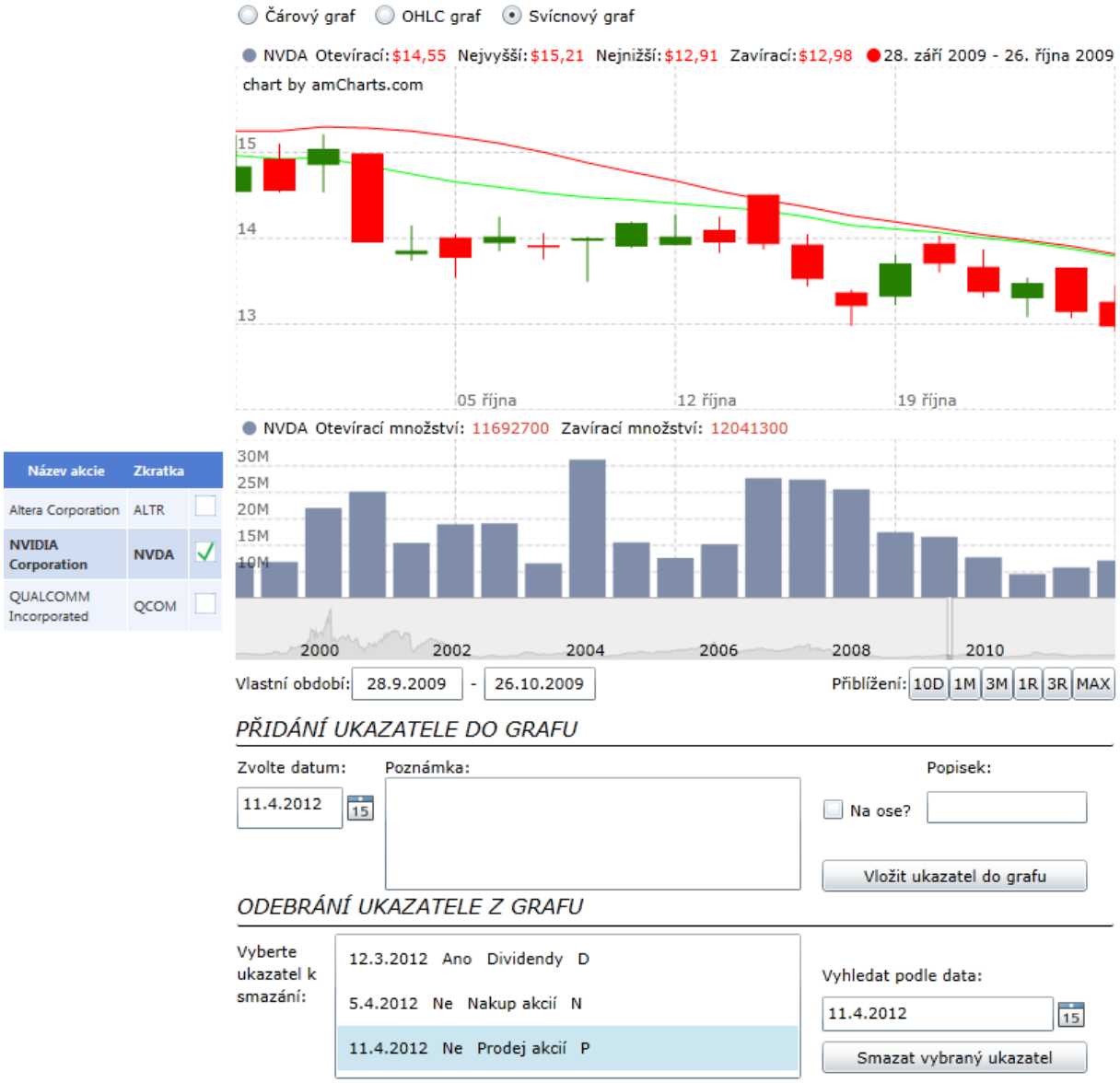
Na levé straně má uživatel možnost vybrat si akcii, pro kterou chce zobrazit grafy. Jakmile si vybere, vykreslí se mu první ze tří grafů a to graf čárový (modrá čára). Všimněte si, že jsou zde ještě další dvě čáry v grafu a to zelená a červená. Tyto čáry reprezentují technické ukazatele. Červená čára zobrazuje SMA a zelená čára zobrazuje EMA. Pod hlavním grafem se nachází sloupcový graf, který zobrazuje množství zobchodovaných akcií. Pod sloupcovým grafem se nachází posuvník a možnost volby zobrazované periody. Nakonec se zde nacházejí dva formuláře. První formulář, s názvem Přidání ukazatele do grafu, má za úkol umožnit přidání uživatelského ukazatele do grafu. Uživatel zvolí datum, ke kterému se má ukazatel vztahovat, poté vepíše text, který má ukazatel po najetí myší zobrazit a nakonec zvolí, zdali se má ukazatel zobrazovat na ose či přímo na čáře grafu, opatří ukazatel popiskem o délce jednoho znaku a vloží ukazatel do grafu. Jak tyto ukazatele v grafu vypadají společně s druhým typem grafu a to grafem OHLC naleznete na obrázku (Obr. 7.8).



Obr. 7.8 Stránka s grafy – druhý typ grafu a uživatelské ukazatele

Samozřejmostí je možnost odstranění ukazatelů z grafu. Ve formuláři Odebrání ukazatele z grafu můžeme zvolit ukazatel pro odebrání. Pokud má ovšem uživatel ukazatelů v grafu více, je zde možnost vyhledání ukazatelů dle zvoleného data. Zvolený ukazatel poté může uživatel smazat.

Obrázek (Obr. 7.9) následně zachycuje poslední typ grafu a to graf svícnový.



Obr. 7.9 Stránka s grafy – třetí typ grafu

Tímto jsme si přiblížili celkovou funkcionalitu prvního z klientů a to webového klienta. Veškerá vývojářská dokumentace této webové aplikace včetně okomentovaných zdrojových kódů je součástí přiloženého disku CD.

8 Implementace klientské části – Mobilní klient

V této kapitole si přiblížíme implementaci druhé klientské aplikace a to mobilního klienta. Shrneme si zde nejdůležitější využité třídy a poté si blíže ukážeme samotnou aplikaci včetně jejího vzhledu a jednotlivých funkcí.

8.1 Použité třídy a komponenty

Aplikace pro operační systém Windows Phone mohou být postaveny na technologii Silverlight či XNA jak zde již bylo řečeno. Moje aplikace využívá první zmíněnou technologii, tudíž pro definici uživatelského rozhraní je využito XAML souborů, kde opět ke každému XAML souboru existuje soubor obsahující funkcionalitu. Kromě toho zde je ještě několik dalších tříd a poté komponenta, která se stará o vykreslování grafů. Aplikace používá pro kontaktování databáze webové služby, hostované webovým klientem. Nyní si zde přiblížíme jednotlivé části.

8.1.1 Třídy starající se o funkcionalitu uživatelského rozhraní

Jednotlivé obrazovky uživatelského rozhraní jsou na Windows Phone známy jako stránky. Těchto stránek je několik různých typů. Moje aplikace využívá jak standardní stránky, tak i tzv. pivot stránky. Na standartní stránce se může uživatel pohybovat nahoru a dolů podobně jako na webové stránce. Na pivot stránkách se lze pohybovat taktéž doleva a doprava, kdy pokud máme například dvě pivot položky v pivot stránce probíhá pohybem vlevo či vpravo výměna těchto položek. Tímto je zajištěna přehlednost a rychlost prohlížení. V aplikaci se nacházejí tyto XAML soubory:

- App
- MainPage
- Menu
- PridatAkcii
- Akcie
- Grafy
- Nastaveni
- OdebratUkazatel
- PridatUkazatel

Každý XAML soubor má přidružen soubor obsahující stejnojmennou třídu a který má stejný název, jen přidává příponu .cs (tedy např. App.xaml.cs a nachází se v něm třída App). App je hlavní XAML soubor aplikace, používá se k definování zdrojů aplikace, sám o sobě nedefinuje žádné GUI. Klíčový je ovšem soubor se třídou přidruženou k tomuto XAML souboru. Kromě automaticky vygenerovaných metod a proměnných obsahuje taktéž metody pro práci s nastavením (ukládání nastavení a načítání nastavení) a taktéž nese veškerá data spojená s chodem aplikace.

MainPage definuje uživatelské rozhraní pro přihlašování uživatelů a přidružený soubor obsahuje třídu obsahující metody nutné pro kontaktování webové služby poskytující ověření.

Menu definuje tří prvkovou pivot stránku. První prvek obsahuje základní informace o aplikaci, druhý prvek zobrazuje seznam akcií přihlášeného uživatele a třetí prvek obsahuje nápovědu.

Přidružená třída poté obsahuje kompletní funkcionalitu této stránky. Jsou zde metody pro kontaktování databáze a načtení, smazání a aktualizaci akcií v portfoliu.

PridatAkcii definuje uživatelské rozhraní pro přidání akcie do portfolia. Přidružená třída se stará o funkcionalitu jednotlivých ovládacích prvků a obsahuje metody pro načtení všech názvů akcií z databáze, jejich uložení do seznamu a zobrazení. Je zde také metoda pro vyhledávání v tomto seznamu akcií a metody pro uložení akcie přidané do portfolia do databáze. Před samotným uložením dojde samozřejmě k ověření, zdali akcie již v portfoliu uživatele neexistuje.

Akcie definuje opět pivot stránku, tentokrát ovšem o čtyřech prvcích. První prvek zobrazuje podrobnější informace o akcii, druhý obsahuje formulář pro přidání uživatelských nákupů či prodejů do databáze, třetí obsahuje seznam nákupů a prodejů akcií dané firmy a nakonec čtvrtý obsahuje údaje o držených akciích, zisku a možnost přepočtu dle kurzu. Přidružená třída se stará o funkcionalitu jednotlivých ovládacích prvků a také obsahuje metody umožňující načtení podrobnějších dat o akcii z databáze, metody pro ověření a přidání uživatelských dat do databáze, metody pro načtení všech uživatelských dat dané akcie z databáze, metody pro ověření a odstranění uživatelských dat z databáze a nakonec metody pro konverzi načtených dat, přepočet hodnot dle zadaného kurzu a výpočet zisku.

Grafy definují stránku, která obsahuje komponentu pro vykreslování grafů. Přidružená třída obsahuje metody pro načtení historických dat a dat uživatelských ukazatelů z databáze. Jakmile jsou všechna data stažena, je zde metoda pro předání těchto dat grafové komponentě a nastavení grafové komponenty.

Nastavení definuje uživatelské rozhraní nastavení technických ukazatelů. Přidružená třída poté opět obsahuje funkcionalitu jednotlivých ovládacích prvků a má na starost také uložení nastavení do paměti.

OdebratUkazatel definuje uživatelské rozhraní pro odebrání uživatelských ukazatelů z grafu. Přidružená třída obsahuje metody pro načtení všech uživatelských ukazatelů z databáze a mazání uživatelských ukazatelů z databáze.

Posledním XAML souborem je PridatUkazatel. Uživatelské rozhraní, které definuje, obsahuje formulář pro přidání uživatelského ukazatele do databáze. Přidružená třída se stará o funkcionalitu ovládacích prvků, obsahuje metody pro ověření přidávaného ukazatele, dále metody pro vložení ukazatele do databáze a znovunačtení seznamu ukazatelů z databáze v případě úspěchu vložení ukazatele.

8.1.2 Ostatní třídy

Kromě výše popsaných tříd starajících se o funkcionalitu daných stránek, jsou zde ještě tyto třídy:

- Barveni
- LoginData
- NastaveniGrafu
- UzivatelskaDataRozsirene

Třída Barveni se stará o konverzi hodnot (nejen, jak by název mohl napovídat, na barevné hodnoty).

Třída LoginData reprezentuje přihlašovací údaje uživatele a je klíčová, neboť instance této třídy obsahuje Cookie Container obsahující přihlašovací informaci, kterou je potřeba předávat webové službě při každé komunikaci.

Třída NastaveniGrafu reprezentuje nastavení technických ukazatelů v grafu. Tato třída je serializovatelná, tedy lze její instanci uložit do tzv. IsolatedStorage, což je úložiště používané Silverlightem.

Třída UzivatelskaDataRozsirene reprezentují uživatelská data (nákupy či prodeje) s rozšířením o proměnnou, která reprezentuje celkovou útratu či zisk.

8.1.3 Komponenta grafu a její třídy

Vzhledem k neexistenci žádné grafové komponenty, která by splňovala požadavky mé aplikace, jsem byl nucen vytvořit komponentu vlastní. Tato komponenta podporuje vše, co umí komponenta ve webové aplikaci. Tedy umí vykreslit všechny tři typy grafů, podporuje zobrazení sloupcového grafu pod hlavním grafem, umožňuje vykreslit uživatelské ukazatele do grafu a taktéž podporuje vykreslení ukazatelů technických (tedy SMA a EMA).

Komponenta samotná se skládá z XAML souboru Graphs, definujícího rozdělení jednotlivých oblastí, do kterých se poté budou vykreslovat grafy, osy atd. Přidružená třída dále obsahuje metodu pro inicializaci grafu, dále metody starající se o vykreslování jednotlivých grafů (čárový, OHLC, svícnový a sloupcový) a jejich os (osa x a osa y). Následně metody pro vykreslování obou typů uživatelských ukazatelů včetně jejich popisků a metody pro vykreslování technických ukazatelů včetně legendy. Taktéž jsou zde metody, které reagují na uživatelskou interakci (klepnutí prstem na displej, dvojité poklepání prstem, posun prstem).

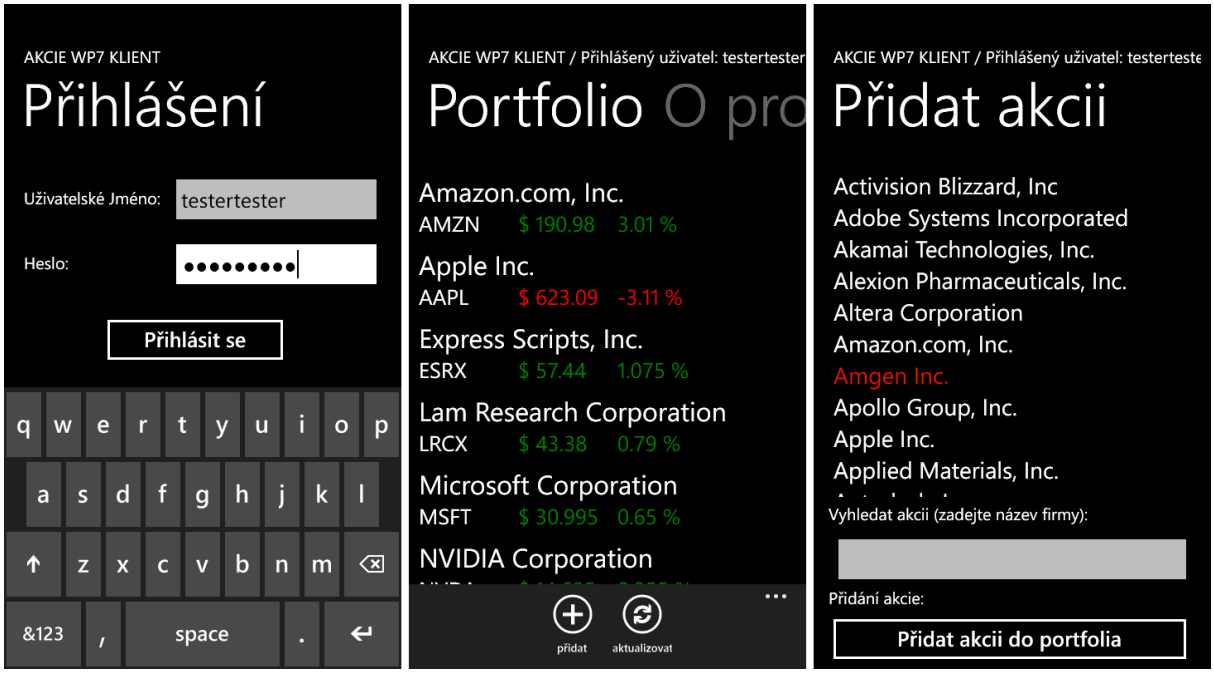
Pro předzpracování dat pro samotnou komponentu slouží třída AnalyzaDat, která obsahuje metody pro zjištění rozsahu hodnot (z důvodu přepočtu číselných hodnot na hodnoty, kterým daná čísla budou odpovídat na obrazovce). Dále metody, které z historických dat vypočítají hodnoty pro vykreslení technických ukazatelů SMA a EMA. A nakonec metoda, která se stará o rozdělení hodnot pro vykreslení popisků osy Y.

Poslední důležitou třídou je třída GrafBodData, která slouží k reprezentaci pozice na grafu.

Tímto jsme se dostali na konec této podkapitoly. Aplikace ještě kromě výše popsaného využívá Windows Phone Toolkit, který přidává podporu dalších komponent uživatelského rozhraní. Toolkit je k dispozici na stránkách Codeplexu a je volně šiřitelný. Moje aplikace využívá komponentu DatePicker, pomocí které se vybírá datum ze seznamu, dále komponentu GestureListener, která se stará o detekování dotykových gest a ContextMenu, která se stará o zobrazení kontextového menu po přidržení prstu například na položce v seznamu.

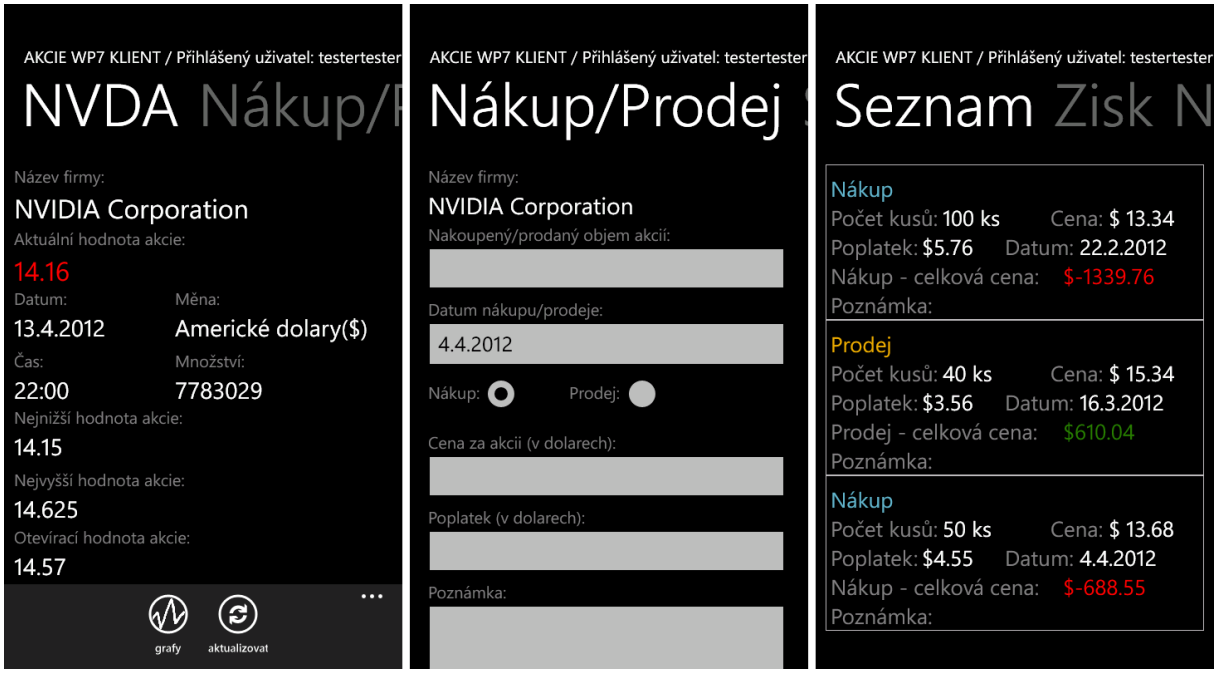
8.2 Mobilní aplikace a její funkce

Nyní se dostáváme již k samotné mobilní aplikaci a její funkcionalitě. Po spuštění aplikace na mobilním telefonu se nejprve zobrazí přihlašovací obrazovku (Obr. 8.1 vlevo).



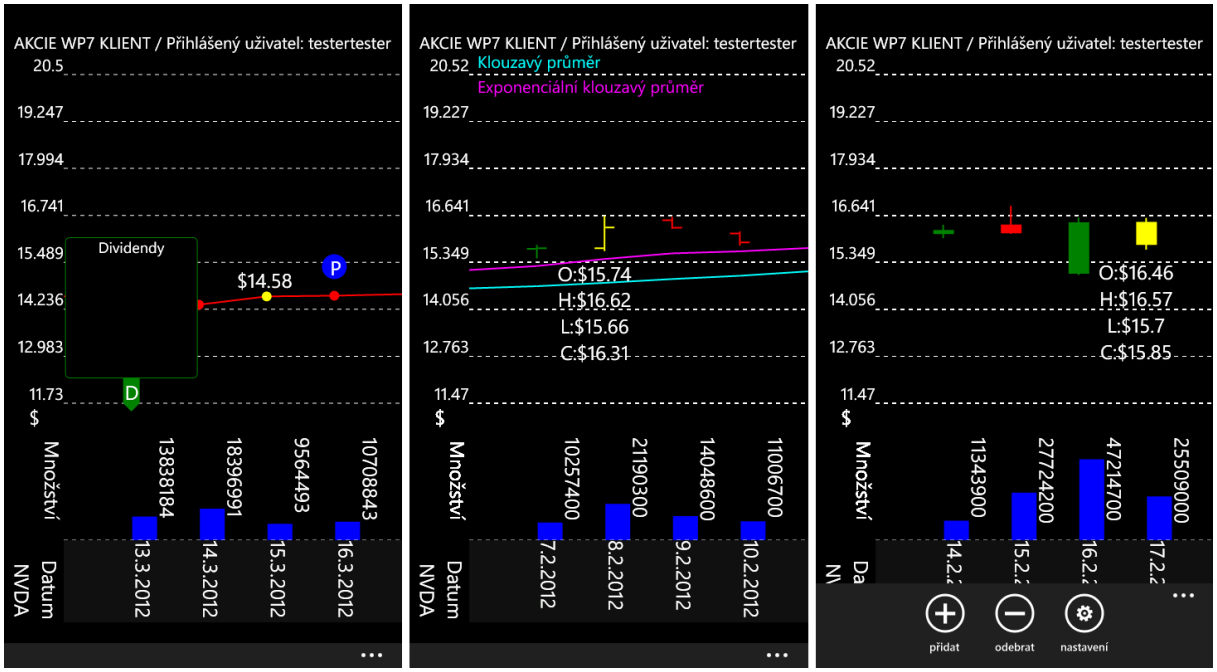
Obr. 8.1 Vlevo přihlašovací obrazovka, uprostřed uživatelské portfolio, vpravo obrazovka pro přidání akcie do portfolia

Na této obrazovce musí uživatel zadat přihlašovací údaje. Pokud údaje ještě nemá, musí se nejprve zaregistrovat přes webovou aplikaci. Jakmile toto udělá, zadá přihlašovací údaje a potvrdí. V případě, že vše zadal správně a je jeho mobilní telefon připojen k internetu, bude přesměrován na titulní stranu. Zde se nachází stručná nápověda jak začít a co dělat. Pokud posune prstem po obrazovce doprava, zobrazí se mu informace o aplikaci. Pokud posune prstem po obrazovce doleva, dostane se na obrazovku, kde může vidět svoje akcie (Obr. 8.1 uprostřed). Poklepáním na akcii se uživatel dostane do upřesňujícího zobrazení (Obr. 8.2 vlevo). Pokud uživatel podrží prst dlouze na dané akcii, objeví se kontextové menu s možností smazání akcie z portfolia. Dále zde může nechat akcie aktualizovat kliknutím na příslušné tlačítko, kdy se načtou aktuální hodnoty daných akcií, popřípadě může přidat další akcie do svého portfolia kliknutím na tlačítko přidat. Jakmile klikne na tlačítko přidat, dostane se na obrazovku pro přidání akcií (Obr. 8.1 vpravo). Zde vybere akcii ze seznamu, popřípadě může nechat akcii vyhledat a poté vybrat a kliknutím na tlačítko Přidat akcii do portfolia tuto akcii přidá do svého portfolia.



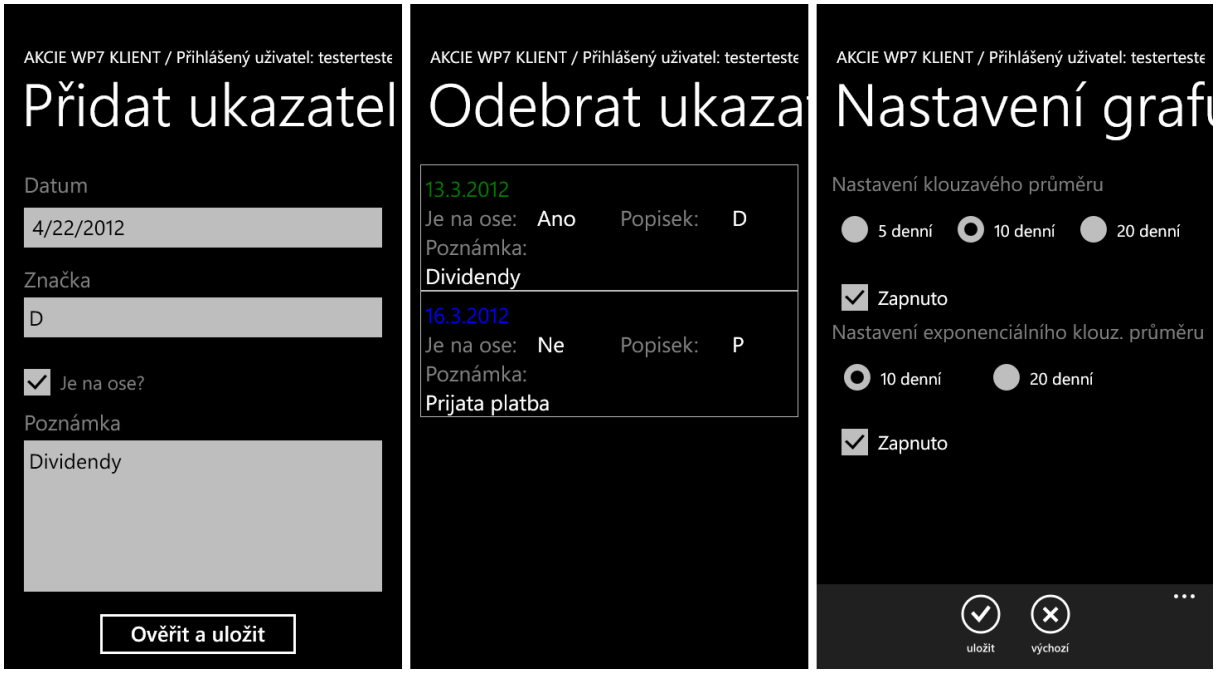
Obr. 8.2 Vlevo obrazovka s upřesňujícími informacemi, uprostřed formulář pro přidání uskutečněného obchodu a vpravo seznam uskutečněných obchodů

Upřesňující zobrazení akcie (Obr. 8.2 vlevo) obsahuje další informace o dané akcii. Kliknutím na tlačítko aktualizovat se aktualizují informace o zobrazované akcii. Pokud uživatel klikne na tlačítko grafy, dostane se do zobrazení grafů (Obr. 8.3). Pokud uživatel posune prstem po obrazovce směrem doleva, zobrazí se mu formulář pro přidání nákupu či prodeje akcií dané firmy (Obr. 8.2 uprostřed). Zde si uživatel může zadat informace, o tom kolik akcií nakoupil, ve který den, za jakou cenu a s jakým poplatkem a opatřit tuto informaci poznámkou. Jakmile potvrdí přidání, dojde k ověření, a pokud vše proběhne v pořádku, bude daný obchod přidán na seznam. Na seznam uskutečněných obchodů (Obr. 8.2 vpravo) se poté uživatel dostane posunutím prstem doleva na obrazovce s formulářem pro přidání obchodu. Zde může přidržením prstu na vybraném obchodu vyvolat kontextovou nabídku a daný obchod smazat. Samozřejmostí je ověření toho, co uživatel maže, aby nedošlo k tomu, že bude mít více prodaných akcií v seznamu než nakoupených. Pokud se ze seznamu opět posune doleva, dostane se na obrazovku, kde je shrnutí. Zde je zobrazen počet prodaných a nakoupených akcií a celkový počet držených akcií. Dále se zde nachází celkové výdaje či příjmy za uskutečněné obchody a také celková hodnota držených akcií vůči aktuální ceně. Uživatel si zde může taktéž přepočítat celkové výdaje či příjmy a celkovou hodnotu držených akcií pomocí libovolného zadaného kurzu.



Obr. 8.3 Vlevo čárový graf s ukazateli, uprostřed OHLC graf se zobrazenými technickými ukazateli a jejich legendou, vpravo svícnový graf se zobrazeným menu

Na obrázku (Obr. 8.3) se nacházejí již výše zmíněné grafy. Aplikace umí vykreslit tři typy grafů, kdy přepínaní mezi nimi, probíhá dvojitým poklepáním na plochu grafu. Na obrázku (Obr. 8.3 vlevo) si můžete všimnout čárového grafu, ve kterém jsou zobrazeny uživatelské ukazatele, a pod samotným grafem je zobrazen objem obchodovaných akcií. Klepnutím na ukazatel lze zobrazit či zavřít poznámku, kterou ukazatel nese. Poklepáním na bod, který leží na červené lince grafu, se zobrazí hodnota akcie daného data. Na obrázku (Obr. 8.3 uprostřed) je zobrazen druhý typ grafu a to graf OHLC. Společně s tímto grafem jsou zde vykresleny taktéž technické ukazatele SMA a EMA a jejich legenda. Legenda se dá zobrazit či skrýt poklepáním na osu y. Opět je zde umožněno zobrazit si hodnoty jednotlivých prvků v grafu poklepáním na ně. Nakonec na obrázku (Obr. 8.3 vpravo) se nachází poslední typ grafu a to graf svícnový. Opět lze poklepáním zobrazit hodnotu prvku. Navíc je zde zobrazeno menu, pomocí kterého se uživatel může dostat na obrazovku pro přidání nového uživatelského ukazatele do grafu, dále na obrazovku pro odebrání uživatelského ukazatele a nakonec do menu nastavení, kde lze nastavit zobrazení jednotlivých ukazatelů SMA a EMA (Obr. 8.4).



Obr. 8.4 Vlevo formulář pro přidání uživatelského ukazatele do grafu, uprostřed seznam ukazatelů pro odebrání, vpravo nastavení technických ukazatelů SMA a EMA

Na obrázku (Obr. 8.4 vlevo) je zobrazen formulář pro přidání uživatelského ukazatele do grafu. Zde po vyplnění a stisknutí tlačítka Ověřit a uložit dojde nejprve k ověření, zdali ukazatel v dané datum již neexistuje, a také k ověření jestli se v daný den obchodovalo. Pokud vše proběhne v pořádku, ukazatel bude přidán. Na obrázku (Obr. 8.4 uprostřed) se nachází seznam všech uživatelských ukazatelů pro danou akcii. Uživatel může přidržením prstu na dané položce vyvolat kontextové menu s možností smazání ukazatele. Nakonec na obrázku (Obr. 8.4 vpravo) je zobrazeno nastavení technických ukazatelů SMA a EMA. Uživatel si vybere, kolikadenní SMA a EMA ukazatel chce a jestli se má daný ukazatel zobrazit. Po kliknutí na tlačítko uložit se změny uloží a při návratu do grafu budou tyto technické ukazatele zobrazeny (pokud je samozřejmě uživatel zapnul). V případě, že uživatel opouští aplikaci a odhlašuje se, jsou změny uloženy přímo do paměti přístroje a opět načteny při dalším přihlášení.

V této kapitole jsme si přiblížili jednotlivé třídy a využité komponenty a taktéž jsme se podívali na funkcionalitu aplikace. Nakonec bych měl ještě dodat, že veškerá data jsou vždy stahována z internetu kvůli aktuálnosti a tedy mobilní zařízení musí být stále v režimu připojeno. Neplatí to pouze pro grafy, jejichž data jsou po jednom načtení uložena v paměti po celou dobu přihlášení.

9 Nasazení a testování

V této kapitole si popíšeme nasazení aplikací a zajištění jejich bezpečnosti a poté si povíme něco o testování a shrneme si zde výsledky.

9.1 Nasazení aplikací

Společně se serverovou aplikací byla na serveru hostována také webová aplikace. Pro běh serveru byl využit standardní počítač s Windows 7 Professional a nainstalovaným IIS serverem verze 7.5, který je nutný pro provoz webových aplikací postavených na technologii ASP.NET. IIS server ve svém výchozím nastavení neumožňuje využití ASP.NET 4.0, takže bylo nutno tuto podporu doinstalovat. Parametry testovacího stroje byly následující:

- Procesor: Core 2 Duo E6850 3GHz@3,6GHz
- Operační paměť: 6GB DDR2 800MHz
- Disk: SSD 128GB Kingston HyperX

Počítač byl připojen k síti internet rychlostí 50MBit/s pro download a 5MBit/s pro upload. V případě serverové aplikace, stačí program spustit, vybrat databázi pro připojení, nastavit zdroje dat a intervaly aktualizace a nechat běžet na pozadí. V případě webového a mobilního klienta je to trochu složitější a budu se zde tomu věnovat v následujících dvou podkapitolách.

9.1.1 Nasazení s výchozím nastavením

Ještě před nasazením je potřeba udělat několik změn, aby vše fungovalo tak jak má. Jednak je potřeba nejprve na MS SQL Serveru vytvořit nový účet korespondující s ASP.NET účtem v IIS, jinak by neměla webová aplikace přístup na MS SQL Server a tudíž by nefungovala korektně. Dále je potřeba editovat soubor Web.config (jedná se o XML soubor), který je součástí webové aplikace a ve kterém je veškeré nastavení webové aplikace (tedy řetězec pro připojení k databázi, nastavení autentizace a autorizace, nastavení membership providera, nastavení klíčů pro šifrování, nastavení webových služeb apod.). Některé údaje vygeneruje Visual Studio 2010 automaticky, některé je potřeba přidat ručně. Všechny údaje jsou opatřeny komentářem v samotném web.configu a nebudu je zde tedy dále rozebírat. Ukážeme si zde pouze údaje, které je potřeba změnit v případě reálného nasazení. Jedinou věcí, kterou je zde nutno změnit, je řetězec pro připojení k MS SQL databázi (tedy v případě, že vývoj probíhal na jiném MS SQL Serveru, než na jakém bude aplikace nasazena). Ten v případě nasazení na testovací sestavu vypadá následovně (Obr. 9.1).

```
<connectionStrings>
  <add name="ApplicationServices"
       connectionString="Data Source=.\SQLEXPRESS;Initial Catalog=AKCIE_DB;Integrated Security=true;" />
</connectionStrings>
```

Obr. 9.1 Nastavení řetězce pro připojení k MS SQL databázi

Toto je jediná úprava, která je nutná pro webového klienta (poté je ještě vhodné změnit machine key, který je zde napevno zadán, a to z důvodu bezpečnosti, ale není to podmínkou). Dále je potřeba nastavit korektní cesty pro Silverlight grafy a mobilního klienta. V projektu Grafy naleznete soubor ServiceReferences.ClientConfig. Tento soubor je generován automaticky při napojení projektu

na webové služby a obsahuje nastavení webových služeb. Zde je potřeba upravit webovou adresu koncového bodu (endpoint). Pokud byla aplikace testována na vývojářském serveru, bude zde nastaveno localhost a číslo portu (Obr. 9.2).

```
<configuration>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="BasicHttpBinding_ISilverlightGraphs" maxBufferSize="2147483647"
                    maxReceivedMessageSize="2147483647">
                    <security mode="None" />
                </binding>
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost:6623/WebService/SilverlightGraphs.svc"
            binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_ISilverlightGraphs"
            contract="GrafyHistorie.ISilverlightGraphs" name="BasicHttpBinding_ISilverlightGraphs" />
          </client>
    </system.serviceModel>
</configuration>
```

Obr. 9.2 Obsah souboru ServiceReferences.ClientConfig

Adresu zde stačí změnit z localhost:6623 na veřejnou IP adresu nebo na doménové jméno, na kterém bude webový klient poté dostupný. Popřípadě můžete ještě přidat podsložku. Stejný postup je potřeba zopakovat u souboru ServiceReferences.ClientConfig v projektu mobilního klienta. Pokud byste toto neudělali, webový klient včetně grafů by fungoval, nicméně mobilní klient by nebyl funkční. Na obrázku (Obr. 9.3) je zobrazen samotný konfigurační soubor mobilního klienta.

```
<configuration>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="BasicHttpBinding_AuthenticationService" maxBufferSize="2147483647"
                         enableHttpCookieContainer="true" maxReceivedMessageSize="2147483647">
                    <security mode="None" />
                </binding>
                <binding name="BasicHttpBinding_IWindowsPhoneService" maxBufferSize="2147483647"
                         enableHttpCookieContainer="true" maxReceivedMessageSize="2147483647">
                    <security mode="None" />
                </binding>
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost:6623/WebService/WindowsPhoneAuthentication.svc"
                binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_AuthenticationService"
                contract="AspNetReference2.AuthenticationService" name="BasicHttpBinding_AuthenticationService" />
            <endpoint address="http://localhost:6623/WebService/WindowsPhoneService.svc"
                binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IWindowsPhoneService"
                contract="AspNetReference1.IWindowsPhoneService" name="BasicHttpBinding_IWindowsPhoneService" />
        </client>
    </system.serviceModel>
</configuration>
```

Obr. 9.3 Obsah souboru ServiceReferences.ClientConfig v projektu mobilního klienta

Opět tedy nahradíme localhost:6623 jako v předchozím případě. Tuto ukázku jsem zde dal ale také z jiného důvodu. Všimněte si parametru enableHttpCookieContainer. Je to právě ten parametr, který zapíná Cookie Container. Cookie Container zde již byl zmiňován, a je velmi důležitý, neboť díky němu jsou přenášeny přihlašovací údaje uživatele.

Jakmile jste se dostali až sem, stačí vše uložit a přenést webovou aplikaci na IIS server. Poté by už neměl být problém dostat se na webového klienta zadáním příslušné adresy a taktéž používat mobilního klienta. V případě testovacího stroje, probíhal přístup k webové aplikaci přes veřejnou IP adresu. Vzhledem k tomu, že počítač byl za směrovačem, bylo potřeba na směrovači nastavit směrování portu 80 na port 80 a privátní IP adresu počítače.

S tímto nastavením byly provedeny všechny testy. Musím zde ale upozornit, že přihlašování v případě mobilního zařízení není nikterak zabezpečeno a pokud se využije software pro odchytávání síťového provozu (např. Wireshark), lze zjistit přihlašovací jméno i heslo uživatele. Samozřejmě, že zabezpečení bylo testováno a v případě webového klienta a jeho grafů bylo taktéž zprovozněno. Problém nastal až u mobilního klienta, jehož nastavení nebylo špatné, ale nerozuměl si s testovacím certifikátem. V následující podkapitole si tedy ukážeme jak nastavit webového a mobilního klienta tak, aby byl přístup k nim zabezpečen.

9.1.2 Nasazení s nastavením zabezpečeného přenosu

Pro zabezpečení zde budeme využívat SSL (Secure Sockets Layer). Jedná se o vrstvu, která se v síťové hierarchii nachází mezi vrstvou transportní a aplikační a poskytuje zabezpečení komunikace šifrováním a autentizací spolu komunikujících stran [21].

Nejprve proveďte veškerá nastavení zmíněná v předchozí podkapitole. Nyní se k těmto nastavením vrátíme. Začneme opět souborem web.config. Kromě úprav zmíněných v předchozí podkapitole je potřeba upravit nastavení autentizace pomocí formulářů a to tak, že změníte parametr requireSSL na true (Obr. 9.4).

```
<!--Nastavení autentizace pomocí formulářů-->
<authentication mode="Forms">
  <forms name=".ASPXAUTH" loginUrl="~/Membership/Login.aspx" protection="All"
         timeout="2880" path="/" requireSSL="true" slidingExpiration="true" cookieless="UseCookies" />
</authentication>
```

Obr. 9.4 Web.config nastavení autentizace pomocí formulářů

Kromě tohoto nastavení je potřeba upravit nastavení samotných webových služeb v souboru web.config. Po úpravě bude nastavení vypadat jako na obrázku (Obr. 9.5).

```
<!--Nastavení webových služeb-->
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="ZabezpeceniPrenosu">
        <security mode="Transport" />
      </binding>
    </basicHttpBinding>
  </bindings>
  <services>
    <service name="System.Web.ApplicationServices.AuthenticationService"
             behaviorConfiguration="AuthenticationServiceBehaviors">
      <endpoint contract="System.Web.ApplicationServices.AuthenticationService"
                binding="basicHttpBinding" bindingConfiguration="ZabezpeceniPrenosu" />
    </service>
    <service name="AkcieWebKlient.WebService.WindowsPhoneService"
             behaviorConfiguration="AuthenticationServiceBehaviors">
      <endpoint contract="AkcieWebKlient.WebService.IWindowsPhoneService"
                binding="basicHttpBinding" bindingConfiguration="ZabezpeceniPrenosu"  />
    </service>
    <service name="AkcieWebKlient.WebService.SilverlightGraphs"
             behaviorConfiguration="AuthenticationServiceBehaviors">
      <endpoint contract="AkcieWebKlient.WebService.ISilverlightGraphs"
                binding="basicHttpBinding" bindingConfiguration="ZabezpeceniPrenosu"  />
    </service>
  </services>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true" multipleSiteBindingsEnabled="true" />
  <behaviors>
    <serviceBehaviors>
      <behavior name="AuthenticationServiceBehaviors">
        <serviceMetadata httpsGetEnabled="true" />
      </behavior>
    </serviceBehaviors>
  </behaviors>
</system.serviceModel>
<system.web.extensions>
  <scripting>
    <webServices>
      <authenticationService enabled="true" requireSSL="true" />
    </webServices>
  </scripting>
</system.web.extensions>
```

Obr. 9.5 Web.config nastavení webových služeb

Tímto jsme zapnuli zabezpečení na straně webové aplikace hostující dané webové služby. Můžete si všimnout, že v nastavení jednotlivých služeb je použit basicHttpBinding. Je to z důvodu, že Windows Phone samotný nepodporuje jinou možnost bindingu.

Nyní je potřeba zapnout toto zabezpečení také u mobilního klienta a grafů v Silverlightu. To je již ovšem velice jednoduché. V souboru ServiceReference.ClientConfig je potřeba změnit security mode z none na nastavení Transport. Poté je potřeba změnit adresu koncového bodu a to tak, že místo http v adrese použijete https.

Tímto jsme dokončili samotnou konfiguraci aplikací. Před samotným nasazením je ještě potřeba zapnout SSL na IIS serveru. K tomu aby SSL fungovalo, je zapotřebí certifikát. A zde je právě onen problém. Pro testovací účely lze vytvořit tzv. self-signed certifikát. Jak lze tento certifikát vytvořit a nastavit IIS pro jeho použití je blíže popsáno zde [22]. Webová aplikace po nasazení funguje bez problémů. Samotné webové služby fungují taktéž, takže jejich nastavení je v pořádku. Problém je na straně operačního systému Windows Phone 7, který není schopen s tímto certifikátem

ani po instalaci do zařízení pracovat. Při reálném nasazení by to ale nebyl žádný problém, protože jakmile by byl využit certifikát od některé z hlavních certifikačních autorit, tyto problémy by odpadly a vše by fungovalo.

Tímto jsme si ukázali jak nastavit aplikaci pro reálný provoz a to jak nezabezpečeně přes standardní http tak i za použití zabezpečeného přenosu přes https. Testování aplikací probíhalo z výše uvedených důvodů pomocí nezabezpečeného provozu, ale jak jsme si již řekli, pokud by se jednalo o ostrý provoz, nebyl by problém jej zabezpečit.

9.2 Testování aplikací

Aby bylo možno odstranit veškeré chyby a vyladit aplikace, bylo nutno je otestovat. Testování probíhalo nejprve pro každou aplikaci zvlášť a poté byly otestovány všechny 3 aplikace zároveň. Nyní se tedy podívejme, jak celé testování probíhalo.

9.2.1 Testování serverové aplikace

Testování aplikace probíhalo na dvou různých testovacích strojích. Parametry prvního stroje zde již byly zmíněny. Jedná se o stejnou sestavu, která byla zmíněna v podkapitole nasazení. Druhý testovací stroj měl poté tyto parametry (jednalo se o notebook):

- Procesor: Core i5 460M 2,53GHz
- Operační paměť: 4GB DDR3 1066MHz
- Disk: SSD 180GB Corsair Force 3

Cílem bylo ověřit, zdali plnění databáze bude probíhat spolehlivě a nebude docházet k zahlcení operační paměti.

Plnění databáze dat v reálném čase pro 100 akcií trvalo 15 až 25 sekund. Rychlejší bylo vždy na první testovací sestavě, která byla připojena ke směrovači kabelem. V případě bezdrátového připojení byly výsledky vždy o něco málo horší. Záleželo ale také na nastavení druhé testovací sestavy. Nejhorší výsledky (22 až 25 sekund) byly dosaženy při režimu úspory energie (kdy procesor běží na frekvenci 1,2GHz). Nicméně v ostrém provozu na serveru by byly podmínky srovnatelné či lepší než v případě první testovací sestavy. Důležité bylo také otestovat, jak se bude aplikace chovat v případě, že bude akcií více (bylo testováno s 500 akciemi v databázi) a interval aktualizace bude nastaven na nejnižší možnou hodnotu a to na jednu minutu. Poté bude totiž docházet k tomu, že než se data akcií stáhnou a uloží do databáze, dojde k zahájení nového stahování. Aplikace si i s tímto na obou testovacích sestavách poradila, sice občas došlo k tomu, že se data některé z akcií nestáhla, ale hned chvíli poté byla stažena znovu a uložena do databáze. Rychlost plnění dat 500 akcií byla zhruba 5-6 násobek rychlosti plnění databáze pro 100 akcií. Pro stahování většího množství dat akcií v reálném čase je ale doporučeno volit vyšší intervaly aktualizace (blíže popsáno je to v nápovědě samotné aplikace).

Plnění databáze v případě historických dat probíhá vždy jednou denně, takže zde nejsou problémy s časem stahování. Byly zde ovšem problémy s využitou pamětí. Nakonec jsem to vyřešil tak, že nejprve se akcie rozdělí do skupin po deseti a poté dojde po každém stažení všech dat pro

oněch 10 akcií k naplnění databáze a pokračuje se na dalších 10 akcií, dokud se nestáhnou data akcií ve všech skupinách.

Tímto jsme si zde shrnuli problémy objevené při testování a jejich následné řešení. Pro otestování reálného provozu jsem vyzkoušel aplikaci nechat běžet celý den. Využití operační paměti při tomto testu nikdy nepřesáhlo 100MB, což bylo cílem vývoje. Největší zatížení je vždy při prvním plnění databáze, kdy je potřeba stáhnout historická data pár let (či desítek let) zpět (okolo 70MB využité paměti). Většinou se pak pohybuje tato hodnota níže, a pokud se v daný čas stahují jen data v reálném čase tak i pod hranící 20MB.

9.2.2 Testování webového klienta

Do testování aplikace bylo přizváno asi 20 lidí. IIS 7.5 na systému Windows 7 limituje maximální počet zároveň vykonávaných požadavků na hodnotu 10. Což se nakonec neukázalo býti problémem, neboť uživatelé nebyli přihlášeni všichni najednou.

Uživatelé měli za úkol zaregistrovat se a poté doslova různě klikat a zkoušet zdali vše funguje jak má. Při testování nenalezli žádné závažné chyby a ty malé byly mnou ihned odstraněny. Byla taktéž otestována funkčnost v případě, kdy uživatel otevře aplikaci ve více oknech, kdy v jednom například něco smaže, ve druhém se tato změna neprojeví a on se bude snažit danou položku smazat znovu. I tyto věci aplikace ustála bez problémů.

9.2.3 Testování mobilního klienta

V případě mobilního klienta byl jediným problémem nedostatek testovacích zařízení. Systém Windows Phone 7 u nás zatím totiž není natolik rozšířen, navíc je potřeba mít telefon odemknut pro testování aplikací. Měl jsem možnost testovat na těchto dvou zařízeních:

- LG E900 Optimus 7
- Samsung SGH-i917 Focus

Obě zařízení patří k první generaci telefonů s tímto systémem. Díky tomu jsou jejich parametry horší než u aktuálně nabízených zařízení. Uživatel se nejdříve zaregistroval přes webového klienta a následně přihlásil pomocí klientské aplikace instalované v telefonu. Poté se již testovala samotná funkcionalita a bezproblémovost aplikace. Osobně jsem testoval také schopnost aplikace pracovat při extrémně slabém mobilním signálu bez dostupnosti mobilní sítě třetí generace. Stahování dat, i přes toto omezení, probíhalo v řádech sekund. Nejdéle trvalo poté stažení dat, které byly nutné jako podklady pro vykreslení grafů. Zde to při slabém signálu trvalo přibližně půl minuty. V případě úplného výpadku sítě aplikace nenačetla data, ale běžela dále a nespadla (bez dostupnosti datové sítě je ovšem aplikace nepoužitelná, neboť nezobrazuje žádná data).

Tímto jsme se dostali na závěr této kapitoly. Všechny aplikace tedy byly úspěšně nasazeny a otestovány. Během testování nebyly nalezeny žádné kritické chyby a všechny ostatní nalezené problémy byly opraveny. To samozřejmě neznamená, že musí být aplikace nutně bezchybná, většina dalších menších chyb by byla nalezena až dlouhodobým nasazením a testováním.

10 Závěr

Výsledkem této diplomové práce je ucelený systém klientských aplikací (webové a mobilní) s vazbou na server, který poskytuje databázi plněnou aktuálními daty z akciových burz. Díky této koncepci se klientské aplikace staly nezávislé na zdroji veřejně dostupných dat. Uživatelské rozhraní všech aplikací je navíc v českém jazyce a aplikace by měly být dostupné zdarma. Myslím si tedy, že zadání práce bylo splněno.

Samozřejmě existuje řada možností, jak aplikace do budoucna vylepšovat, a díky obsažené kompletní vývojářské dokumentaci a plně okomentovanému zdrojovému kódu, jsou tyto možnosti výrazně zjednodušeny. Kromě vylepšování aktuálních aplikací, by bylo možno vytvořit mobilní klienty na další operační systémy mobilních zařízení jako například Android od firmy Google či iOS od firmy Apple, což by mohly být zajímavá témata bakalářských prací. Jako vedlejší produkt při vývoji poté vznikla knihovna pro vykreslování grafů pro systém Windows Phone 7, která by s trochou úprav mohla poskytovat vše potřebné i pro jiné aplikace nejen tohoto zaměření.

Použitá literatura

[1] ACHELIS, Steven B. Technical analysis from A to Z: covers every trading tool-- from the Absolute Breadth Index to the Zig Zag. Chicago: Probus Pub., c1995, 331 s. ISBN 15-573-8816-4.

[2] NESNÍDAL, Tomáš a Petr PODHAJSKÝ. Obchodování na komoditních trzích: průvodce spekulanta. 1. vyd. Praha: Grada, 2005, s. 55. ISBN 80-247-1499-X.

[3] Moving Averages: Simple and Exponential. Stockcharts [online]. 14.10.2011 [cit. 2012-04-13]. Dostupné z: http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_averages

[4] NAGEL, Christian, Bill EVJEN, Jay GLYNN, Morgan SKINNER a Karli WATSON. C# 2008: Programujeme profesionálně. Vyd. 1. Brno: Computer Press, 2009, s. 30-31. ISBN 978-80-251-2401-7.

[5] The .NET Framework Stack. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-04-24]. Dostupné z: http://en.wikipedia.org/wiki/File:DotNet.svg

[6] .NET Framework Conceptual Overview. MICROSOFT CORPORATION. MSDN [online]. 14.9.2010 [cit. 2012-04-12]. Dostupné z: http://msdn.microsoft.com/en-us/library/zw4w595w.aspx

[7] NASH, Trey. Accelerated C# 2010. New York: Distributed to the book trade worldwide by Springer Science Business Media, c2010, 627 s. Expert's voice in C#. ISBN 14-302-2538-6.

[8] MACDONALD, Matthew. Pro .NET 2.0 Windows Forms and custom controls in C?: [learn how to use and extend the Windows Forms toolkit to create anything from a vector-drawing program to a document-view framework]. New ed. Berkeley, Calif: Apress, 2006. ISBN 15-905-9439-8.

[9] NAGEL, Christian, Bill EVJEN, Jay GLYNN, Morgan SKINNER a Karli WATSON. C# 2008: programujeme profesionálně. Vyd. 1. Brno: Computer Press, 2009, s. 479-511. ISBN 978-80-251-2401-7.

[10] CHAPPELL, David. Introducing Windows Communication Foundation in .NET Framework 4. MICROSOFT CORPORATION. *MSDN* [online]. 2010 [cit. 2012-04-24]. Dostupné z: http://msdn.microsoft.com/library/ee958158.aspx

[11] PÍSEK, Slavoj. ASP.NET: začínáme programovat : podrobný průvodce začínajícího uživatele. 1. vyd. Praha: Grada, 2003, 228 s. ISBN 80-247-0526-5.

[12] MACKEY, Alex a Stefan TURALSKI. Introducing .NET 4.0 with Visual Studio 2010. Berkeley, Calif.: Apress, 2010, s. 97-114. ISBN 978-143-0224-563.

[13] LAIR, Robert. Beginning Silverlight 4 in C♯. New York: Apress, 2010. ISBN 978-143-0229-896.

[14] PETZOLD, Charles. *Programming Windows Phone 7*. Redmond, WA: Microsoft Press, 2010. ISBN 978-073-5643-352.

[15] SCHNEIDER, Robert D. *Microsoft SQLserver 2008 all-in-one desk reference for dummies(r)*. 1st ed. Indianapolis, IN: Wiley Pub., Inc., 2008. ISBN 04-701-7954-6.

[16] RANDOLPH, Nick. *Professional Visual Studio 2010*. Indianapolis, IN: Wiley Pub., c2010, 1177 s. Wrox professional guides. ISBN 04-705-4865-7.

[17] VB.NET: Jak napsat CAPTCHA komponentu pro ASP.NET?. HERCEG, Tomáš. *VB.NET* [online]. 20.9.2009 [cit. 2012-04-21]. Dostupné z: http://www.vbnet.cz/clanek--138-tipy_a_triky_pro_asp_net_dil_1_jak_napsat_captcha_komponentu_pro_asp_net_.aspx

[18] AmCharts for wpf and silverlight. AMCHARTS. *AmCharts* [online]. 2012 [cit. 2012-04-24]. Dostupné z: http://wpf.amcharts.com/stock

[19] Implementing a Membership Provider. MICROSOFT CORPORATION. *MSDN* [online]. 2011 [cit. 2012-04-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/f1kyba5e%28v=vs.85%29.aspx

[20] How to: Sample Membership Provider Implementation. MICROSOFT CORPORATION. *MSDN* [online]. 2011 [cit. 2012-04-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/6tc47t75.aspx

[21] Secure Socket Layer. ONYSZKO, Tomasz. *WindowsSecurity.com* [online]. 2002, 2004 [cit. 2012-04-25]. Dostupné z: http://www.windowsecurity.com/articles/secure_socket_layer.html

[22] How to Create a Self Signed Certificate in IIS 7. *SSLShopper* [online]. 2010 [cit. 2012-04-24]. Dostupné z: http://www.sslshopper.com/article-how-to-create-a-self-signed-certificate-in-iis-7.html

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha A: Adresářová struktura přiloženého CD

/Aplikace	Obsahuje všechny tři aplikace a soubor s instrukcemi.
/Dokumentace	Vývojářská dokumentace všech obsažených aplikací.
/Projekt	Kompletní Visual Studio 2010 projekty všech aplikací. Obsahuje všechny zdrojové kódy.
/Text	Text diplomové práce ve formátu .docx a .pdf